Ricardo João Cruz Correia

Integração de Gráficos Vectoriais e Informação Estruturada em Ambientes Distribuídos



Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
1998

Ricardo João Cruz Correia

Integração de Gráficos Vectoriais e Informação Estruturada em Ambientes Distribuídos



Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Ciência de Computadores

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
1998

Conteúdo

Resumo 7

Abstract 8

Índice de Figuras 10

1 Introdução 11

1.1 Objectivo 11

1.2 Abordagem global 13

2 O Estado de Arte 16

2.1 GIS 16

2.1.1 Sistema de gestão de dados 17

2.1.2 Introdução e manipulação de informação geográfica 17

2.1.3 Consulta, análise e visualização sobre dados geográficos 18

2.2 Exemplos de aplicações 21

2.2.1 DOOGIS 21

2.2.2 DrawBase 22

2.2.3 CADD/GIS 22

2.2.4 Building Audit System 23

2.2.5 GIS (NASA) 23

2.2.6 Autodesk MapGuide 24
2.2.7 Autodesk World - Sistema GIS 26
2.2.8 Magma/Lava GIS Browser 27

3 Modelo Conceptual **31**

3.1 Representação da informação 31
3.1.1 O espaço como entidade gráfica 31
3.1.2 O espaço como entidade funcional 32
3.1.3 A interligação das duas formas de ver o mundo 33
3.2 Distribuição da informação 35
3.2.1 JDBC 39
3.2.2 "Serialização" 40

4 Implementação do sistema **44**

4.1 Um objecto BlueBase 44
4.1.1 Áreas 46
4.1.2 Blocos 47
4.1.3 Associação de elementos à base de dados 49
4.2 A utilização de um objecto BlueBase 50
4.3 O utilitário BindData 52
4.3.1 Selecção de dados 52
4.3.2 Processamento gráfico 53
4.3.3 Identificação das áreas 55
4.3.4 Localização de linhas 61
4.3.5 Associação de áreas à base de dados 62
4.3.6 Associação de blocos à base de dados 64

4.3.7 Criação de caminhos entre áreas 65
4.3.8 Gravar o objecto criado 65
4.4 Detalhes de implementação 67
4.4.1 Tabelas de partição 67
4.4.2 O formato DXF 68

5 Aplicações do Modelo Construído **70**
5.1 Espaço interior de um edifício 71
5.1.1 Navegar na planta de um edifício 71
5.1.2 Gerir um edifício 72
5.1.3 Avaliar um projecto 74
5.2 Edifícios de uma área urbana 75
5.2.1 Consultar informação de recenseamentos de uma cidade 75
5.2.2 Visualizar os espaços para aluguer de uma cidade 75
5.3 Sub-regiões de uma grande região 76

6 Conclusão **78**
6.1 Verificação dos objectivos de desenho 79
6.2 Problemas de implementação 80
6.3 Trabalho futuro 81

A Classes de Objectos **85**
A.1 "package" project 85
A.1.1 Classe BlueBase 85
A.1.2 Classe Property List 89
A.1.3 Classe Area 89
A.1.4 Classe BlockDef 90

A.1.5 Classe Block 91
A.1.6 Classe Path 92
A.1.7 Classe ProjCanvas 92
A.1.8 Classe GraphicsContext 93
A.1.9 Classe DataBaseConnection 94
A.2 "package" geometry 94
A.2.1 Classe FPoint 95
A.2.2 Classe FLine 95
A.2.3 Classe FRectangle 96
A.2.4 Classe FPolygon 96
A.3 "package" quadtree 97
A.3.1 Classe QuadTree 97

Referências **98**

Resumo

O objectivo deste trabalho é integrar informação gráfica vectorial, dados estruturados (ex.: bases de dados) e funcionalidades que actuam sobre a informação armazenada, tendo em vista a sua utilização em ambientes distribuídos via Internet.

A sua estrutura foi direccionada para ser utilizada em aplicações de Gestão de Edifícios, embora possa também ser usada por outras aplicações que necessitem da integração de informação gráfica vectorial com dados estruturados.

A representação é implementada em Java, tirando partido da orientação por objectos, e capacidade de integração em páginas WWW. Da implementação resulta uma classe de objectos que se denomina por **BlueBase**, e que alberga informação gráfica, informação que permite aceder às bases de dados remotas que contêm a informação estruturada, e funcionalidades que interligam os dois tipos de informação.

Como parte do sistema foi criado um utilitário denominado **BindData** que permite a elaboração de objectos BlueBase. Este utilitário utiliza informação gráfica vectorial armazenada em ficheiros no formato DXF.

Os objectos BlueBase pretendem ser interlocutores inteligentes entre as aplicações que os utilizam e as representações gráfica e estruturada que os compõem. Com o intuito de testar os objectos BlueBase e as funcionalidades que possuem, foram criadas aplicações que permitem navegar no interior de um edifício, navegar na planta de uma cidade, e executar consultas às respectivas bases de dados. Estas aplicações acedem aos objectos BlueBase através da WWW.

Abstract

The main goal of this work is to integrate graphical information, structured data (ex.: databases) and functionalities that act on both types of information, aiming its use in a distributed environment through the Internet.

Its structure was initially intended to be used in Facilities Management applications, although it can be used by another type of applications that need to integrate graphical information and structured data.

The representation is implemented in Java, taking advantage of being an object-oriented language, and capable of developing applications that are "nested" in a WWW page. This representation is called **BlueBase**. The structured information is stored in databases. The databases are accessed when applications request the BlueBase objects for structured information.

This project includes an utility program called **BindData** that allows the creation of BlueBase objects. This utility program uses DXF files to read vector drawings into the system.

The BlueBase objects are intended to be intelligent interfaces between the applications that use them and the graphical representation and structured data that composes them. Applications were created that allow to display a building blueprint and to display a city blueprint, and to execute queries to the respective databases, with the intention of testing the functionalities of the BlueBase objects. These applications access to the BlueBase objects through the WWW.

Lista de Figuras

1.1 Representação da classe BlueBase. 13

2.1 Interface da aplicação Autodesk World - Sistema GIS. 28

2.2 Interface do "applet" Lava/Magma. 30

3.1 Fases de criação do objecto da classe BlueBase. 34

3.2 Associação de uma representação vectorial a um registo de base de dados. 35

3.3 Elementos da criação do objecto da classe BlueBase. 35

4.1 Utilização objecto da classe BlueBase. 45

4.2 Estrutura de um objecto BlueBase. 46

4.3 Representação de espaços e portas. 48

4.4 Representação dos caminhos existentes entre espaços. 48

4.5 Exemplos de tipos de blocos. 49

4.6 Associação de um objecto área a um registo através do campo chave. . 49

4.7 Esquema de acesso ao sistema. 51

4.8 Esquema de utilização do sistema. 51

4.9 Interface gráfico do utilitário BindData. 52

4.10 Interface para escolha de camadas do DXF. 54

4.11 Interface para escolha do número de vértice da linearização de um círculo. 55

4.12 Execução do algoritmo geométrico. 56

4.13 Encontra a primeira linha. 57
4.14 Encontra o primeiro vértice. 58
4.15 Cálculo de ângulos para encontrar o primeiro vértice. 58
4.16 Cálculo de ângulos para encontrar novo vértice. 59
4.17 Segmentos que não "cortam" o limite do espaço. 59
4.18 Segmentos desnessários à detecção do limite do espaço. 59
4.19 Árvore simples da "quadtree". 61
4.20 Imagem sucessivamente dividida pela "quadtree". 62
4.21 Exemplo da "quadtree" utilizada. 62
4.22 Espaço fechado encontrado. 63
4.23 Selecção de espaços existentes na base de dados. 63
4.24 Exemplo de um espaço destacado. 64
4.25 Blocos na planta. 65
4.26 Criação de caminhos. 66

5.1 Interface do "applet" de navegação em plantas. 73
5.2 Freguesias da cidade do Porto. 77

6.1 Símbolo que identifica uma porta. 81
6.2 Fotografia aérea do S. da Pedra com vectorização de alguns edifícios. . 83

Capítulo 1

Introdução

1.1 Objectivo

Este trabalho tem como objectivo a integração de informação gráfica com informação estruturada. Com a união destas duas formas diferentes de assimilar a realidade pretende-se criar uma única representação que beneficie das características de ambas.

Este tipo de interligação tem sido utilizada em vários campos, como a Gestão de Edifícios, os Sistemas de Informação Geográfica, e aplicações de Integração e Gestão de Informação Georeferenciada. As aplicações pertencentes a estas campos sofrem normalmente de alguns problemas que limitam as potencialidades destas aplicações, nomeadamente:

- Utilização preferencial da imagem do tipo "raster";
- Dificuldade de utilização em ambientes distribuídos;
- Dependência de utilização dos formatos criados em relação a uma plataforma (Computador/Sistema Operativo).

A utilização da imagem de tipo "raster" (não vectorial) não é apropriada para o processamento de figuras geométricas. O cálculo de áreas e perímetros e as transformações geométricas, são algumas possibilidades que o desenho gráfico vectorial tem como vantagem quando comparado com o desenho tipo "raster".

Outra questão actual que se reveste de importância é a distribuição da informação. Os sistemas até agora existentes são normalmente orientados para a execução em estações

de trabalho ou computadores pessoais. Por esta razão são de difícil distribuição numa rede local heterogénea ou na Internet.

A distribuição de dados numa rede em que existem diferentes plataformas (computadores/sistemas operativos), requer a utilização de formatos que sejam independentes dessas plataformas. Para além da distribuição de dados também é abordada, a questão de utilização de programas num ambiente distribuído.

Os objectivos de desenho do sistema foram:

- Formato de dados integrado que reuna ambas as representações (gráfica e estruturada);
- Funcionalidades básicas que manipulam a integração de representações;
- Capacidade de distribuição do formato de dados na Internet;
- Possibilidade de criação de pequenas aplicações com objectivos específicos;
- Incorporação da informação de domínio relativo a edifícios (tipos de espaços).

O sistema implementado cria um formato que reúne informação gráfica e estruturada, relacionando os componentes de cada uma, permitindo encarar o conjunto de toda a informação como uma única entidade.

Esta entidade é implementada segundo um modelo orientado por objectos. Foi definida uma classe de objectos, denominada **BlueBase**, que integra informação gráfica e estrutural, e um conjunto de funcionalidades básicas que actuam sobre os dados interligados. Desta forma, os dados e os métodos que actuam sobre os dados, são distribuídos como um todo (figura 1.1).

Diferentes aplicações podem utilizar a mesma entidade, fazendo uso dos vários métodos disponíveis para atingir os seus objectivos específicos.

O sistema foi desenvolvido para facilitar a representação de informação num domínio relativo a edifícios. Assim, os dados da entidade estão estruturados de acordo com os tipos de espaços existentes no interior de um edifício.

A representação gráfica da entidade é de carácter vectorial, sendo a representação estruturada implementada utilizando um acesso a uma base de dados remota. Para além das características já enunciadas e tendo em atenção a grande diversidade de aplicações que poderiam utilizar o sistema, este deverá ter em atenção os seguintes aspectos:

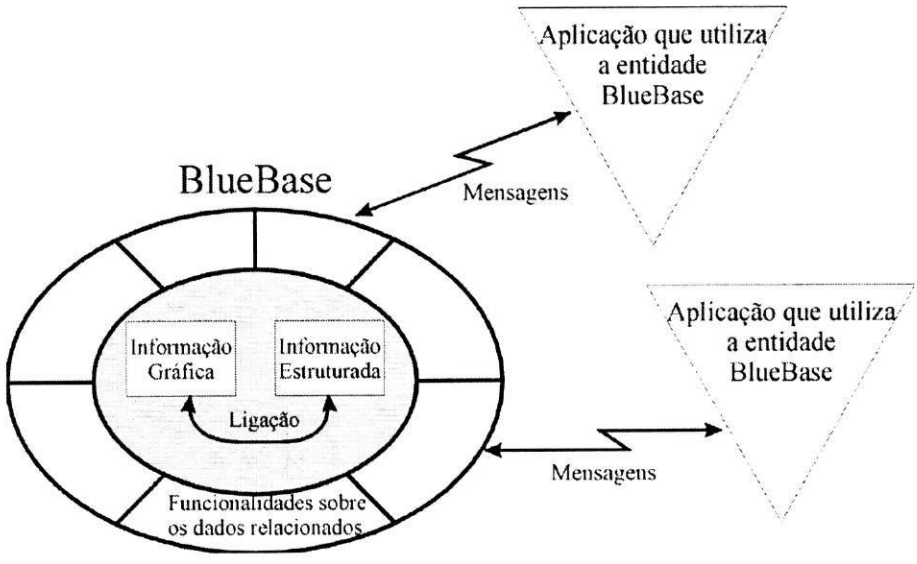


Figura 1.1: Representação da classe BlueBase.

- A utilização de formatos "standard" dos ficheiros que contêm a informação a disponibilizar (gráfica e estruturada) para facilitar a interligação com outras aplicações;
- A utilização de gráficos vectoriais de forma a disponibilizar funções de cálculo geométrico;
- Simplicidade e flexibilidade na utilização das funções que compõem a biblioteca;
- Transparência na utilização de uma rede como meio de comunicação;

1.2 Abordagem global

A classe BlueBase integra entidades que representam salas, corredores, equipamento como computadores, extintores, etc. Estas entidades são representadas como objectos que, por sua vez, contêm a sua representação gráfica e dados estruturados. Sob a forma de métodos são associadas funcionalidades aos objectos que podem ser invocados pelas aplicações.

A integração dos dois tipos de dados (gráficos e estruturados) numa única estrutura, permite aproveitar a facilidade de utilização e a intuitividade de uma representação gráfica, e a capacidade de estruturação e de consulta de uma base de dados, para além de facilitar a sua distribuição numa rede de computadores.

Esta classe de objectos reúne um conjunto de funcionalidades habituais a aplicações que necessitem de um interface que interligue representações gráficas com bases de dados, na qual se incluem as aplicações no campo do GIS.

Os métodos existentes na classe BlueBase tiram partido da ligação existente a informação gráfica e a informação estruturada.

A definição das ligações é efectuada pelo utilitário **BindData**. Este permite ao utilizador definir as entidades que deseja incluir no objecto BlueBase final.

A simplicidade na utilização dos métodos permite ao criador de aplicações (programador) concentrar-se em questões de mais alto nível, dado que a classe de objectos se encarrega das questões de implementação a baixo nível.

Muitas das instituições que poderiam utilizar este sistema, já possuem os dados que pretendem disponibilizar em ficheiros. Assim, é fundamental a reutilização dos dados, quer tenham sido criados propositadamente para o sistema em questão ou não. Esta questão é válida para os ficheiros que contêm representação gráfica tal como para as bases de dados. São utilizados "standards" de comunicação com base de dados de forma utilizar a informação nestas armazenada.

Para a representação gráfica, foi escolhido um formato vectorial em deterioramento dum formato "raster", pelas seguintes vantagens:

- Exactidão no cálculo de propriedades geométricas como a área, perímetro, ponto médio, etc.;
- Possibilidade de efectuar transformações gráficas com mais exactidão, como rotação e ampliação;
- Ficheiros mais pequenos, o que é fundamental para a distribuição numa rede de computadores;
- Maior facilidade na associação entre as entidades existentes na representação gráfica (divisões, material de escritório, escadas, etc.) e as bases de dados.

A título de exemplo serão criadas algumas aplicações com diferentes funcionalidades que fazem uso do referido pacote de objectos.

A primeira das aplicações aqui proposta é composta por uma planta de uma cidade portuguesa associada aos dados que foram recolhidos através do último recenseamento

efectuado à população portuguesa. A criação de uma aplicação que disponibilizasse estes dados poderia usufruir de plantas previamente elaboradas e das bases de dados em formato digital do recenseamento. Seria fácil elaborar um cruzamento de variáveis existentes nos inquéritos realizados à população. O resultado do cruzamento de variáveis, se fosse possível, seria visualizado na representação gráfica. Desta forma, algumas das conclusões seriam obtidas mais rapidamente com a observação do interface gráfico.

Outra aplicação do sistema associa a planta de um edifício a uma base de dados que recolhe informação relativa às actividades que existem no mesmo. Esta aplicação actua como um mapa que nos indica o local onde se dá uma actividade (como uma aula ou uma conferência). Podemos então inquirir o mapa do caminho mais curto até ao local da actividade. É também possível ter acesso aos dados relativos aos utilizadores das instalações (como onde se encontra o escritório do prof. X).

Capítulo 2

O Estado de Arte

Existem vários campos que utilizam representações gráficas interligadas com bases de dados, como GIS, Gestão de Edifícios e Gestão de Informação Georeferenciada. Nenhuma destas categorias é estanque, pelo que existem muitas aplicações que cruzam duas ou mais destas categorias.

2.1 GIS

Um sistema de informação geográfica (GIS) é um sistema que permite a visualização de mapas, análise dos itens existentes e eventos que podem ocorrer. O GIS integra operações comuns de bases de dados como a consulta e a análise estatística com os benefícios da visualização e análise geográfica que os mapas permitem. Estas funcionalidades distinguem o GIS de outros sistemas de informação e tornam este sistema importante para um largo espectro de aplicações que pretendam explicar eventos, prever resultados e planear estratégias.

Um sistema de GIS disponibiliza as funções e ferramentas necessárias ao armazenamento, análise e visualização de informação geográfica. Os principais componentes são:

- Um sistema de gestão de dados;
- As ferramentas para a introdução e manipulação de informação geográfica;
- As ferramentas que suportam consultas, análise e visualização sobre dados geográficos;

2.1.1 Sistema de gestão de dados

O componente mais importante de um sistema GIS são os dados. Para pequenos projectos GIS pode ser suficiente armazenar a informação geográfica como ficheiros simples. Porém, quando o volume de dados torna-se grande e o número de utilizadores aumenta, é normalmente preferível utilizar um sistema de gestão de base de dados (DBMS) para ajudar o sistema no armazenamento, organização e gestão dos dados.

Existem diferentes paradigmas de DBMS, mas para o GIS as bases de dados relacionais (RDBMS) têm sido as mais úteis. No sistema de base de dados relacional, os dados são conceptualmente um conjunto de tabelas. Campos comuns a diferentes tabelas servem para as ligar entre si. Este desenho de sistema surpreendentemente simples tem sido largamente utilizado principalmente devido à sua flexibilidade.

2.1.2 Introdução e manipulação de informação geográfica

Um sistema GIS armazena informação sobre o mundo como uma colecção de camadas temáticas que podem estar interligadas entre si geograficamente. Cada camada representa um tipo de informação, como solo, altitude, fronteiras de países, etc.

A informação geográfica ou contém uma referência geográfica explícita, como a latitude e a longitude, ou uma referência implícita como um endereço, código postal, o nome de uma família, um identificador de um parque florestal, ou um nome de uma rua.

A informação geográfica trabalha fundamentalmente com dois tipos diferentes de modelos geográficos - o modelo vectorial e o modelo "raster". No modelo vectorial, a informação sobre pontos, linhas e polígonos é codificada e armazenada como uma colecção de coordenadas (x, y). A localização de um elemento com as características de um ponto, como um poço, pode ser descrito como uma coordenada (x, y). Elementos lineares, como estradas e rios, podem ser armazenados como um conjunto de coordenadas de pontos. Os elementos com forma de polígono, como áreas de território e edifícios, podem ser armazenados como um conjunto fechado constituído por vários pontos.

O modelo vectorial é extremamente útil para descrever características discretas, mas menos útil para descrever características que variam continuamente como tipos de solo e altitudes. O processo de conversão dos dados de mapas de papel para o computador é usualmente denominado por digitalização. Uma imagem "raster" é composta por uma grelha de pontos que resultam da digitalização de um mapa ou de uma fotografia.

Ambos os modelos, vectorial e "raster", têm vantagens e desvantagens que os tornam apropriados para diferentes objectivos. Os sistemas de GIS modernos são capazes de trabalhar com ambos os modelos.

2.1.3 Consulta, análise e visualização sobre dados geográficos

Os sistemas standard GIS executam essencialmente cinco processos ou tarefas:

- Introdução;
- Manipulação;
- Gestão;
- Consulta (análise);
- Visualização.

Antes da informação geográfica poder ser utilizada no GIS, os dados são convertidos para um formato digital apropriado. Os sistemas de GIS modernos podem automatizar completamente o processo de digitalização. Actualmente, muitos tipos de dados geográficos já existem em formatos compatíveis com os sistemas GIS.

É provável que os tipos de dados necessários para um projecto GIS em particular tenham que ser transformados ou manipulados de alguma forma para os tornar compatíveis com o sistema. Por exemplo, a informação geográfica é disponibilizada em diferentes escalas (ficheiros com detalhes de ruas; áreas menos detalhados para a execução de censos; e níveis de regiões por código postal). Antes desta informação poder ser integrada, tem de ser transformada para a mesma escala (grau e detalhe de pormenor). Pode ser uma transformação temporária apenas por questões de visualização ou uma transformação permanente necessária para a análise. A tecnologia GIS oferece muitas ferramentas para a manipulação de dados espaciais e para filtrar dados desnecessários.

Após existir um sistema GIS em funcionamento que contém a informação geográfica necessária, é possível elaborar perguntas simples como:

- Quem é o dono de uma dada parcela de terreno?
- Qual é a distância entre dois locais?

É também possível fazerem-se perguntas analíticas como:

- Quais são os locais apropriados para a construção de novas casas?
- Qual é o tipo de solo predominante nas florestas de pinheiros?
- Se eu construir um auto-estrada aqui, como será o tráfico afectado?

O GIS está relacionado directamente com outros tipos de sistemas de informação, mas é a capacidade de manipular e armazenar dados geográficos que distingue a tecnologia GIS. Apesar de não existir regras rígidas e simples para classificar os sistemas de informação, a discussão seguinte deverá ajudar a diferenciar o GIS de:

- "Desktop mapping";
- Desenho assistido por computador (Computer Aided Design);
- "Remote sensing".

Um sistema do tipo "desktop mapping" utiliza um mapa para organizar os dados e como interface com o utilizador. Estes sistemas focam-se na criação de mapas: o mapa é a base de dados. A maioria deste tipo de sistemas têm limitações quanto à gestão de dados, análise espacial e personalização das funcionalidades. Estes sistemas operam em computadores pessoais como PCs, Macintoshes, e pequenas workstations UNIX.

Os sistemas CAD são capazes de criar desenhos e plantas de edifícios e de infra-estruturas. Esta actividade necessita que componentes de características fixas sejam montados de forma a criar uma estrutura global. Estes sistemas têm poucas regras quanto à forma como estes componentes são montados e possuem capacidades de análise muito limitadas. Os sistemas CAD foram estendidos para suportar mapas mas têm utilidade limitada na manutenção e análise de grandes bases de dados geográficas. Estes sistemas produzem uma representação vectorial.

"Remote sensing" é a disciplina de executar medições da terra utilizando sensores como câmaras colocadas em aviões, receptores GPS (sistemas de tecnologia de posicionamento global), e outros dispositivos. Estes sensores coleccionam dados na forma de imagens e permitem a manipulação, análise e visualização das imagens. Porém como não suportam uma gestão eficiente de dados geográficos e de operações de análise, não podem ser chamados de sistemas GIS.

Os sistemas GIS possuem capacidades simples como apontar e clicar. Também possuem ferramentas sofisticadas de análise que informam periodicamente gestores e analistas. Mas, a tecnologia GIS torna-se realmente poderosa quando é levada a executar análise de dados geográficos na procura de padrões e de possíveis cenários. Uma dessas ferramentas, a análise de proximidade, permite que sejam respondidas questões da seguinte forma:

- Quantas casas existem a menos de 100m desta conduta de água?
- Qual é o número de clientes a 10km desta loja comercial?
- Que proporção da cultura de alface está a 500m do poço?

Para que seja possível fazer este tipo de análise é necessário que exista a capacidade de trabalhar com diferentes tipos de dados.

A integração de diferentes níveis de dados (layers) envolve um processo designado por "overlaying". De forma simplificada, este poderia ser pensado como uma operação visual, mas as operações analíticas necessitam que um ou mais níveis de dados sejam fundidos fisicamente. Este processo pode, por exemplo, integrar dados do solo, elevação, vegetação com o custo das terras.

Em muitos tipos de operações geográficas o resultado final é melhor visualizado como um mapa ou gráfico. Os mapas são muito eficientes para armazenar e comunicar informação geográfica. O GIS criou um conjunto de novas ferramentas para a cartografia. Os mapas podem ser integrados em relatórios, vistas tridimensionais, imagens fotográficas, e outros meios visuais como multimédia.

Projectos mais recentes têm dado um maior realce à manutenção de informação relativa ao interior das instalações. Desta forma podem ser mantidos dados relativos a instalações eléctricas, condutas de ar, redes de computador, distribuição de espaços por vários departamentos, etc. Para este tipo de aplicações tornou-se fundamental utilizar outro tipo de interface gráfico. São agora frequentemente utilizados gráficos vectoriais de plantas na substituição de plantas do tipo "raster". Estas permitem uma maior correcção no cálculo de características numéricas (como áreas e perímetros) e também permitem uma maior eficiência na localização de entidades na planta.

Já existem algumas aplicações comerciais deste tipo de produtos. Estes produtos têm normalmente uma de duas falhas. Ou a quantidade de dados que permitem disponibilizar é limitada, ou são aplicações "monoposto" que não permitem o acesso via ambiente distribuído (Intranet ou Internet).

Outra questão importante prende-se com o facto de muitas destas aplicações não visarem a utilização do público em geral, mas apenas de gestores de edifícios.

Uma das principais preocupações dos projectos mais recentes prende-se com o interface gráfico utilizado para questionar a base de dados, uma questão muitas vezes negligenciada.

2.2 Exemplos de aplicações

O estudo que se segue pretende dar uma panorâmica geral das aplicações comerciais e dos sistemas que estão em fase de investigação.

De seguida passo a descrever alguns projectos importantes que definem o nível existente nesta área de aplicações.

2.2.1 DOOGIS

Este projecto é importante na medida em que é inovador na utilização da Internet e da linguagem de programação Java na área do GIS. A ideia do DOOGIS (Dynamic Object Oriented Geographic Information System) baseia-se na concepção de que a Internet é uma grande base de dados e, utilizando a tecnologia associada a objectos, podem-se criar objectos espaciais inteligentes. Os objectos espaciais podem residir em qualquer parte na Internet. Estes podem ser acedidos por um Applet Java como se tivessem numa única base de dados coerente localizada no seu computador. Os objectos podem ser descarregados para um browser de Internet (mais particularmente para um Applet de Java) como objectos completos com dados e comportamentos (objectos inteligentes).

A versão disponível na Internet é a versão 2.0 e está ligada a um servidor de dados espaciais escrito na linguagem Smalltalk. Esta versão demonstra as capacidades básicas do GIS como execução de zoom e deslocamento, para além de permitir aceder a informação que reside na World Wide Web. Esta também demostra a capacidade de criação de objectos dentro do applet e o envio dos mesmos para o servidor, neste caso, objectos que representam endereços de páginas na Internet (URLs). Este projecto está a ser desenvolvido por Argonne National Laboratory.

2.2.2 DrawBase

O Drawbase(R) é uma aplicação comercial, que visa a gestão de edifícios.

A aplicação é configurada na instituição que pretende gerir as instalações, fazendo uso dos ficheiros CAD existentes. Estes desenhos e outros dados que se queiram gerir serão utilizados na criação de ficheiros do tipo Drawbase(R). Os ficheiros criados armazenam os desenhos inteligentes já anteriormente referidos.

Este sistema já foi implementado na West Virginia University, com uma base de dados que contém dados sobre 400 diferentes instalações da mesma universidade. A aplicação permite aos utilizadores seleccionar edifícios através do mapa e ter acesso aos dados a estes associados.

Os gestores das instalações utilizam a aplicação para procurar números de quartos, áreas de espaços, utilização de espaços, localização de material perigoso, redes de comunicação e instalações eléctricas.

Este projecto está a ser desenvolvido por Design Alliance e CADworks(R), Inc.

Encontra-se disponível em `www.lm.com/~tda`

2.2.3 CADD/GIS

O Tri-Service CADD (GIS Technology Center) está neste momento a desenvolver formatos de dados standard para sistemas de informação geográfica (GIS) e para CADD (computer-aided design and drafting). Estes formatos de dados visam ser utilizados no planeamento ambiental, arquitectura, engenharia e construção, e gestão de instalações.

Os Tri-Service Spatial Data Standards (TSSDS) foram desenvolvidos como um único modelo de planeamento de estruturas para instalações da Força Aérea, do Exército e da Marinha tal como para alguns projectos de engenharia civil.

O objectivo desta instituição é a criação de suportes para aplicações de GIS funcional, ou seja, aplicações GIS com desenhos capazes de responder a perguntas.

Este projecto está a ser desenvolvido por Tri-Service CADD/GIS Technology Center.

Encontra-se disponível em `fwgcom.wes.army.mil/projects/standard/tssds`

2.2.4 Building Audit System

A aplicação Building Audit System (BAS) foi desenvolvido por Automated Facilities Management Inc. de forma a permitir a criação de relatórios automáticos e de ferramentas de gestão de edifícios. Estas aplicações estão vocacionadas quer para a comunidade de gestão de edifícios quer para a comunidade de aplicações GIS. O software permite uma relação próxima entre uma forma gráfica e uma base de dados. Este programa tem como principal vantagem a facilidade com que se transita da representação gráfica para a base de dados e vice-versa. Uma vez instalado, o programa permite ao utilizador documentar, planear, gerir, e alocar recursos de forma simples. Aos desenhos gráficos podem ser efectuadas perguntas relativamente a áreas e outros valores geométricos.

Este projecto está a ser desenvolvido por Automated Facilities Management.

Encontra-se disponível em `www.interlog.com/~afm`

2.2.5 GIS (NASA)

A NASA (em particular o Langley Research Center Hampton, Virginia) tem um projecto GIS para permitir a gestão de grandes quantidades de informação relativa a várias infra-estruturas. A tecnologia GIS é vista por esta entidade como uma ferramenta poderosa e capaz de produzir decisões intuitivas por parte dos seus utilizadores. Ela é capaz de integrar Sistemas de Gestão de Informação com Sistemas de Informação Gráfica e de visualizar as relações entre ambos.

O projecto da NASA tem como principal alvo a gestão de uma grande quantidade de dados que são neste momento mantidos de forma não integrada. Estes dados têm origem "ambiental" (meteorológica, geográfica, demográfica, etc.) e funcional (dados das instalações a manter).

Relativamente ao hardware e ao "software" necessários para a aplicação, a NASA mantém o projecto em aberto. Desta forma permite que novas evoluções tecnológicas sejam facilmente integradas no seu projecto. Os principais elementos técnicos do sistema incluem: um conjunto de servidores que actuam como recursos de informações, um conjunto de workstations/terminais que funcionam como interface para os utilizadores, uma rede de comunicação para os ligar, os protocolos para permitir a transferência de informação, e o "software" para processar e analisar a informação. Os servidores comunicam através da rede para responder a pedidos de informações

de outros servidores ou a pedidos de workstations utilizando para tal protocolos de comunicação standard. Alguns servidores servem ficheiros via NFS (Network File System) ou FTP (File Transfer Protocol), alguns são capazes de produzir procuras e de aceder a perguntas do tipo SQL, e alguns executam "software" GIS e interagem com o utilizador através de um interface gráfico como "X" ou via um interface de documentos multimédia como o HTTP (HyperText Transfer Protocol). A NASA não põe de parte a hipótese de utilizar outro tipo de protocolos com o objectivo de integrar da melhor forma toda a informação disponível.

Este projecto da NASA teve o seu início em 1992 e prevê-se que continue em fase de análise até o ano de 1999. Deverão ser analisadas quais as bases de dados a incluir, qual o hardware e "software" a incluir e de que forma deverão os utilizadores aceder à informação.

Este projecto está a ser desenvolvido por Langley Research Center Hampton, Virginia (NASA).

Encontra-se disponível em `GIS-www-larc.nasa.gov/bplan`

2.2.6 Autodesk MapGuide

A aplicação Autodesk MapGuide (distribuição de informação georeferenciada em Internet ou intranet) é um produto de uma grande empresa conhecida principalmente pelo seu "software" CAD - a Autodesk.

A aplicação em causa contém um interface direccionado para a criação de mapas. Estes mapas suportam a integração de informação gráfica em formato vectorial e "raster".

Como é habitual neste tipo de aplicações é possível controlar o nível de interactividade que queremos que o mapa tenha com o utilizador. Esta aplicação utiliza outros produtos para aumentar a interactividade com o utilizador. É de realçar que é possível existirem associações com bases de dados relacionais externas ao sistema, ao correio electrónico, e pacotes de "software" capazes de elaborar gráficos.

Os mapas criados por esta aplicação podem ser incorporados em páginas de Internet. É ainda possível colocar controlos em JavaScript(R) ou VBScript(R) na página HTML (formato utilizado na Internet) que interagem com o mapa.

Este sistema suporta múltiplas conexões a ficheiros geográficos e múltiplas RDBMSs (Relational Data Base Management System - Sistema de Gestão de Bases de Dados

Relacionais), como Oracle(R), Sybase(R), e Microsoft Access, residentes localmente ou num sistema operativo UNIX(R) acessível através de uma rede de computadores.

Eis algumas características e funcionalidades desta aplicação:

- Capacidade de funcionamento em intranets e na Internet;
- Importação de muitos formatos de dados GIS;
- Criação de níveis estáticos para acesso rápido a dados;
- Simbologia de mapas temática;
- Capaz de funcionar com 700 diferentes sistemas de coordenadas;
- Associação de objectos no mapa a vídeo, relatórios, documentos e outras aplicações da Web;
- Capaz de alterar a informação vectorial (funciona como programa de desenho).

Para a visualização dos mapas criados existe uma aplicação que actua como cliente do servidor de GIS. Esta aplicação permite a interacção com mapas simples e complexos, estender e colapsar níveis (layers) temáticos, criação de consultas, executar relatório personalizados sobre os objectos seleccionados, imprimir de acordo com uma escala, etc.

Eis uma lista de algumas características e funcionalidades desta aplicação:

- Suporte de formatos "raster" e vectoriais;
- Selecção de objectos por listagens, distâncias, polígonos e intersecção;
- Visibilidade e atributos por escala;
- Legendas interactivas e controlo de níveis;
- Ferramentas de navegação no mapa;
- Criação de marcas (bookmarks) nos mapas;
- Acesso a bases de dados remotas;
- Capaz de funcionar nas plataformas dos browsers da Netscape(R) e Microsoft(R).

2.2.7 Autodesk World - Sistema GIS

Esta aplicação é, ao contrário da aplicação anteriormente apresentada, um sistema GIS puro. Este sistema pode ser descrito em quatro diferentes áreas: acesso, integração, criação e edição, análise e visualização.

Quanto ao acesso aos dados este tem a capacidade de:

- Os formatos de ficheiros nativos da aplicação incluírem: AutoCAD(R) (DWG), Autodesk World Geobase, e Microsoft Access (MDB);
- Capaz de manipular ficheiros em vários diferentes formatos;
- Ligação bidireccional a sistemas de bases de dados Oracle, Sybase, Microsoft SQL Server, ou outros que suportem ODBC/DAO;
- Inserção e modificação de uma vasta gama de formatos de imagens que incluem: JPEG, TIFF, EPS.

A integração tem as seguintes funcionalidades:

- Combinar CAD, GIS, imagens "raster", atributos e ficheiros multimédia num único ambiente;
- Trabalhar com múltiplas tabelas de atributos como uma única tabela "virtual";

Quanto à criação e edição são disponibilizadas as seguintes operações:

- Execução de digitalização;
- Visualizar e editar simultaneamente;
- Ler dados através de GPS;
- Visualizar e geo-referenciar imagens "raster" com a possibilidade de criar, verificar e editar níveis vectoriais sobrepostos.

O processo de análise tem as seguintes características:

- Criação de filtros de selecção para efectuar subconjuntos de dados para análise;

- Execução de operações espaciais, de atributo e de relação (ponto no polígono, linha no polígono, intersecções);
- Execução de operações espaciais como encontrar todas as entidades numa determinada área num ficheiro gráfico;
- Execução de consultas combinando filtros gráficos, espaciais e de bases de dados.

A visualização e apresentação da informação geográfica tem as seguintes características:

- Criação de múltiplas vistas de conjuntos de dados, cada uma com as suas características próprias de visualização;
- Visualização de todos os dados num projecção, mesmo que os dados estejam armazenados em projecções diferentes;
- Criação de uma variedade de visualizações temáticas (por exemplo gráficos de tarte);
- Criação de apresentações com relatórios e gráficos.

Este sistema tem a grande vantagem de integrar com facilidade os sistemas CAD no seu próprio GIS. Este produto permite que os utilizadores tirem vantagem da experiência e produtividade que adquiriram com outros produtos. O interface de base de dados é idêntico ao utilizado pelo Microsoft(R) Access, utiliza o "software" Seagate's Crystal Reports para a criação de relatórios, e a linguagem interna de "script" é Microsoft Visual Basic. Este produto pode aceder à grande quantidade de mapas executados em AutoCAD(R), porque lê e escreve no formato DWG (formato standard do "software" de CAD mais utilizado - AutoCAD(R)).

A título de exemplo a figura 2.1 apresenta o interface da aplicação.

2.2.8 Magma/Lava GIS Browser

A aplicação Magma/Lava GIS browser é um sistema de navegação GIS completamente executado em Java. Este facto leva a que seja independente do hardware e facilmente distribuído em intranets ou na Internet. Este sistema suporta a integração de mapas com origem em diferentes servidores, o acesso a grandes bases de dados, dados em

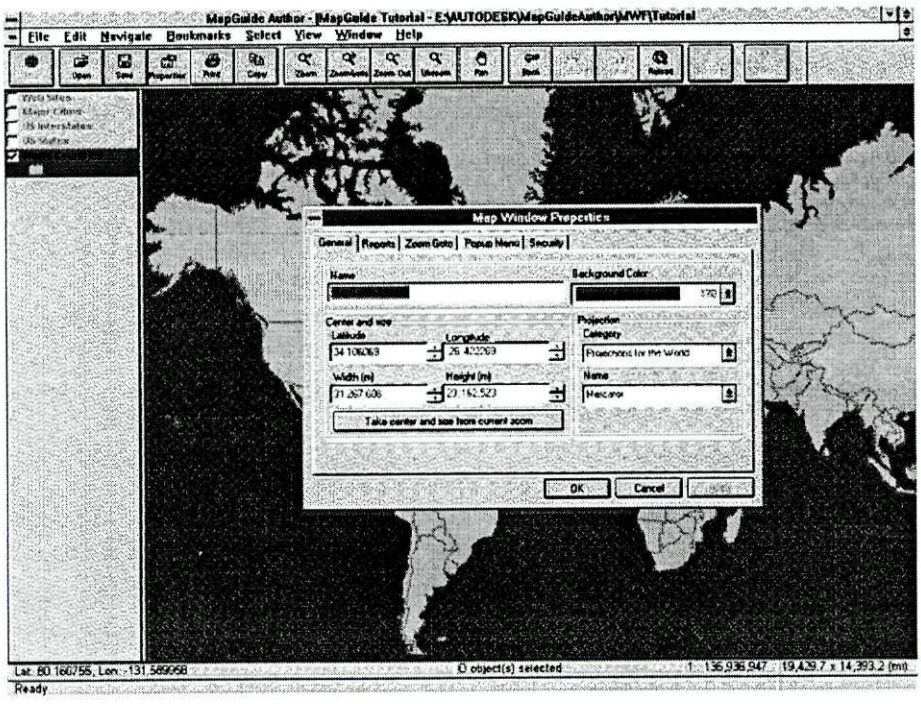


Figura 2.1: Interface da aplicação Autodesk World - Sistema GIS.

formato vectorial e "raster", e sistemas de cache para diminuir a quantidade de tráfego de rede.

A aplicação que serve de interface ao sistema GIS denomina-se Lava. O sistema que actua como servidor de informação geográfica tem o nome de Magma.

Algumas das características especiais desta aplicação são:

- Selecção de dados por atributos;
- Integração de sistema de pagamento para serviços comerciais de GIS;
- Cache local;
- Existência de níveis de acesso aos dados.

O Lava é direccionado para meta-informação, ou seja, são apresentadas ao utilizador um conjunto de operações que foram definidas para um objecto. Estas operações podem incluir a invocação automática de novas aplicações (porque a aplicação é distribuída em browsers falamos de Applets) que existam no fornecedor do serviço.

Este sistema é actualmente suportado pelos browsers capazes de executar aplicações em Java. Em particular: Netscape 3.01 e superior, Microsoft Explorer 3.0 e superior

e HotJava.

Quanto ao servidor de dados - Magma - pode ser combinado com as seguintes sistemas de informação: Informix Universal Server com PGS TopGIS, Illustra, CA OpenIngres (1.2, 2.0), Oracle com SDO, a base de dados espacial associada a PGS.

O Magma recebe pedidos numa linguagem de acesso a base de dados (LQL - limited query language). Os pedidos são traduzidos numa consulta apropriada ao sistema que gere o modelo objecto - base de dados relacional. O resultado é disponibilizado num formulário com um formato standard.

Os seguintes itens são personalizáveis em cada implementação do sistema Magma:

- O modelo de meta-informação que apresenta ao mundo exterior a informação que o Magma disponibiliza;
- Os modelos que traduzem as consultas no formato LQL em consultas à base de dados;
- O modelo comercial de pagamento de serviços: este modelo define a que dados têm acesso grupos de utilizadores.

A título de exemplo a figura 2.2 apresenta o interface do "applet" que se encontra disponível em `www.pgs.nl`.

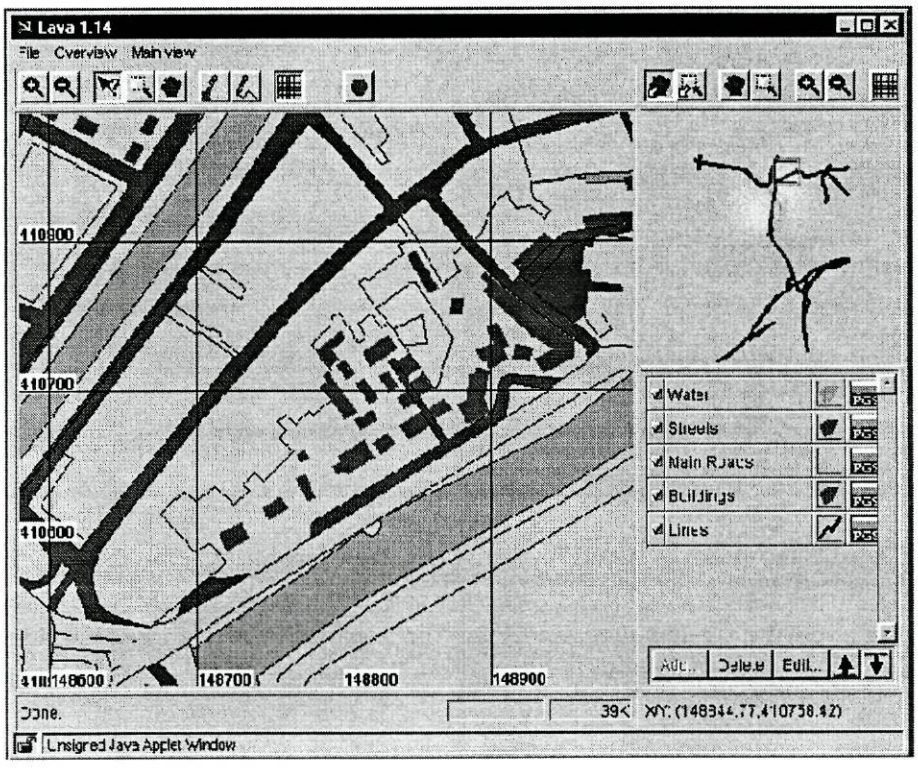


Figura 2.2: Interface do "applet" Lava/Magma.

Capítulo 3

Modelo Conceptual

Neste capítulo é analisada a estrutura conceptual que tem como objectivo agrupar a informação gráfica e estruturada que descreve um edifício e estabelecer ligações entre as duas representações. Os dados interligados destinam-se a ser distribuídos via WWW por forma a serem usados por aplicações finais. Neste capítulo é ainda discutida a forma de implementação de aplicações na Internet que conduziu à escolha da linguagem Java para implementar todo o sistema.

3.1 Representação da informação

Os elementos que se pretendem representar são **espaços** (divisões de edifícios, salas, corredores, átrios, etc.) e **blocos** (equipamento como computadores, extintores, etc.). Estes possuem uma representação gráfica e um conjunto de propriedades. Destes dois tipos de elementos, os espaços são as entidades centrais que se pretendem representar.

3.1.1 O espaço como entidade gráfica

As aplicações de desenho assistido por computador (CAD) utilizam primitivas gráficas a partir das quais se constróem as entidades. Estas primitivas são basicamente pontos, linhas, polígonos, elipses, arcos, entre outras. Através destes elementos básicos é possível representarem-se entidades tão distintas como edifícios, peças mecânicas, sistemas eléctricos, caras de pessoas, animais, etc.

A representação gráfica que se pretende é muito mais específica quando comparada

com a que estas aplicações de desenho nos permitem. Com efeito, apenas pretendemos representar três tipos de entidades: áreas que representam espaços identificados no projecto, caminhos que podem ser percorridos, e blocos básicos que simbolizam o equipamento existente como computadores e extintores. Os objectos mantidos pelo sistema são áreas, caminhos e blocos ao contrário das aplicações de desenho cujos objectos são normalmente, pontos, linhas, polígonos e outros elementos geométricos. Mesmo as aplicações especializadas em desenho técnico (principalmente as aplicações CAD - Computer Aided Design) baseiam a criação de novos desenhos nos elementos geométricos simples, apenas facultando um conjunto de símbolos predefinidos como portas, janelas, louça sanitária, etc.

Esta questão seria de menor importância se não fosse objectivo deste trabalho a utilização de ficheiros "standard" no sistema. As aplicações CAD são sem dúvida uma referência muito forte no que diz respeito à representação gráfica de entidades. Por esta razão foram elaboradas ferramentas que permitem extrair da representação utilizada por alguns sistemas CAD a informação necessária para a criação da representação gráfica que pretendemos.

O processo de identificação dos limites de um espaço é o início da geração de um objecto que o representa. É normal existirem linhas que representam paredes que atravessam vários espaços diferentes. Desta forma é impossível a atribuição destas linhas a apenas um espaço. Os objectos que pretendemos fazer representar devem ser independentes e não fazer depender a sua representação de outros elementos partilhados por outros objectos.

As áreas identificadas são divididas segundo o seguinte critério arquitectónico: áreas de serviço, áreas servidas e áreas de circulação. Esta divisão por funcionalidades é um factor estruturante dos projectos em construção. A atribuição de uma funcionalidade a uma dada área pode ser considerado o primeiro passo na identificação dessa mesma área. É também associado a cada área um nome que a identifica.

3.1.2 O espaço como entidade funcional

A forma gráfica do espaço é apenas uma parte da representação do objecto. A restante parte é composta pelas propriedades não gráficas do objecto.

Por questões de eficiência de utilização, e de aproveitamento de dados já existentes, as propriedades de cada objecto são armazenadas numa base de dados. A cada objecto é associado um registo de uma consulta ou tabela. A cada propriedade do objecto está

associado um campo da mesma consulta ou tabela. Quando inquirida a base de dados retorna as propriedades do registo que está associado ao objecto.

Os dados que se encontram nesta base de dados podem ser mantidos em paralelo com a utilização do sistema. Sempre que forem pedidas ao sistema as características de dado objecto, a base de dados é consultada. Podem existir aplicações de manutenção da base de dados sem que esta questão seja um problema para o sistema. Quando o gestor da base de dados criar novos campos para os objectos, a aplicação tem imediatamente acesso aos mesmos.

A utilização de um registo de uma tabela para armazenar as propriedades de cada objecto é simples e normalmente suficiente. Para ilustrar a flexibilidade desta forma de representar as propriedades de cada objecto suponhamos as duas situações seguintes:

Representação de um edifício de uma faculdade. Cada área é um espaço do edifício. A cada espaço são associados os seguintes campos: departamento a que pertencem, nome do responsável pela sala, nome do professor (no caso de ser um gabinete), tipo de soalho (por questões de limpeza). Esta informação é disponibilizada aos alunos para que a utilizem na procura de determinada sala ou gabinete, tal como para o gestor da manutenção do edifício que inquire o sistema sobre que tipo de soalho existe em cada sala.

Representação da planta de uma cidade. Cada área é um edifício. Este sistema está associado à uma base de dados proveniente de um recenseamento. Os campos são agora o número de pessoas que habita cada edifício, número de crianças, idade do edifício, se possui canalização, se tem saídas de emergência, preço do imóvel, etc. As consultas a esta base de dados poderiam incluir perguntas como:

- Onde se situam os edifícios que têm mais de quarenta anos ?
- Qual o preço médio desses imóveis?

3.1.3 A interligação das duas formas de ver o mundo

As duas formas de representar o objecto espaço (a forma gráfica e por propriedades) não são concorrentes mas sim complementares. Cada uma destas representações tem as suas vantagens. O objectivo deste sistema é permitir que as aplicações criadas à custa do sistema tirem partido do que cada uma das representações tem de melhor.

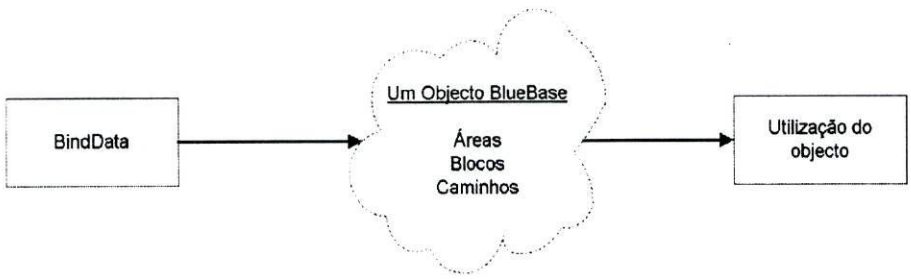


Figura 3.1: Fases de criação do objecto da classe BlueBase.

Para além de ser possível utilizar a representação gráfica como forma de consultar os dados, é de realçar que é também possível utilizá-la como forma de visualizar resultados. O sistema contém métodos capazes de seleccionar espaços com base em critérios de consulta. A resposta neste caso, pode não ser dada sob a forma de uma enumeração dos objectos que respeitam as condições da consulta mas sim sob a forma gráfica no interface. (Quando o sistema for inquirido sobre que salas possuem uma área superior a cem metros quadrados, a resposta poderá ser visualizada destacando na planta os espaços que cumprem o critério).

Este sistema pressupõe a criação de uma estrutura (neste caso um objecto da classe BlueBase) para posterior utilização pelas aplicações finais (figura 3.1). Este objecto é criado através do utilitário **BindData** que utiliza um ficheiro de representação gráfica e uma base de dados. Após criado, o objecto da classe BlueBase pode ser acedido localmente ou através de uma rede de computadores. Cada objecto pode ser utilizado em diferentes aplicações. (A criação de um objecto da classe BlueBase que descreva o edifício de uma faculdade pode ser utilizado numa aplicação que faça a gestão dos espaços existentes, bem como por uma aplicação que funcione como um mapa e que descreva o caminho mais perto entre dois espaços.)

A criação de um objecto da classe BlueBase é iniciado com a selecção da base de dados que contém a informação a disponibilizar, bem como a selecção do ficheiro que contém a representação gráfica.

A segunda fase do processo é composta pela selecção das áreas que fazem partem do modelo que queremos representar e posterior associação a uma entrada da base de dados (figura 3.2). (Após seleccionar um polígono que descreve a "Sala X" na planta do edifício é associado a este polígono um registo de uma base de dados que contém os dados a disponibilizar da respectiva "Sala X".)

Desta forma é possível inquirir a base de dados, utilizando como interface a planta de um edifício. O inverso também é possível, ou seja, após elaborar uma consulta à base

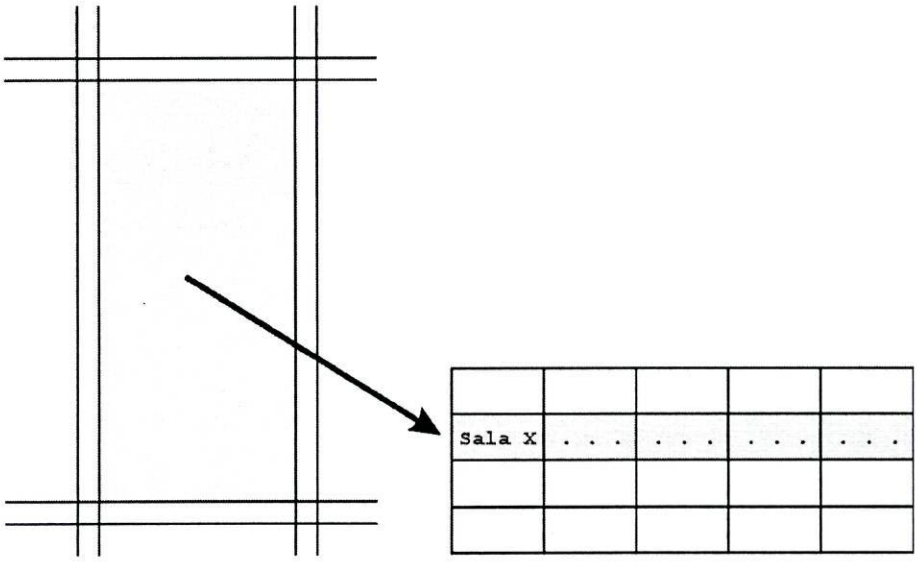


Figura 3.2: Associação de uma representação vectorial a um registo de base de dados.

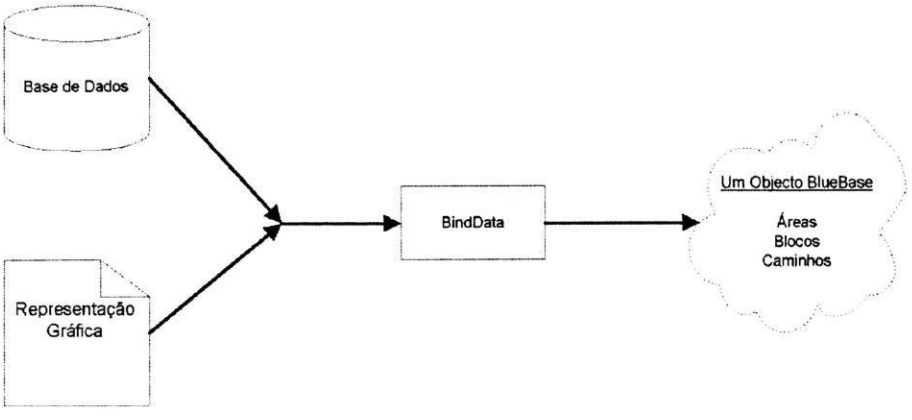


Figura 3.3: Elementos da criação do objecto da classe BlueBase.

de dados é possível a visualização do resultado na planta. (Ao inquirimos a base de dados sobre quais as salas que pertencem a dado departamento, é possível visualizar a resposta na planta.)

É da interligação da representação gráfica e dos dados estruturados na base de dados que resulta o objecto da classe BlueBase (figura 3.3).

3.2 Distribuição da informação

O sistema foi desenvolvido tendo como objectivo uma utilização num ambiente distribuído, apesar de não haver nenhum impedimento nem desvantagem na utilização

deste sistema num ambiente local. Esta é uma das razões pela escolha da linguagem Java na implementação do sistema.

Apesar do Java ter capacidade para resolver os problemas tradicionais que são executados em máquinas locais, a razão pela sua importância é que também resolve problemas de programação na WWW (World Wide Web).

Para se entender o que realmente é a WWW, ajuda voltar um pouco atrás e entender melhor os sistemas cliente/servidor.

A principal ideia de um sistema cliente/servidor é a existência de um repositório central de informação - algum tipo de dados - que se quer distribuir ao ser requisitada por um conjunto de pessoas ou de máquinas. A chave do conceito cliente/servidor é que o repositório de informação está centralizado num único local de forma que seja possível alterá-lo e que as alterações sejam propagadas para os consumidores da informação. O repositório da informação, o "software" que distribui a informação e as máquina(s) onde a informação e o "software" residem formam o servidor. O "software" que reside na máquina remota, e que comunica com o servidor, que faz o pedido da informação que a processa e a visualiza na máquina remota é chamado de cliente.

Sempre que o "software" de cliente é alterado, tem de ser construído, compilado e instalado nas máquinas cliente, o que torna o processo mais complicado e dispendioso do que pode parecer; é especialmente problemático suportar múltiplos tipos de computadores e de sistemas operativos.

Finalmente, existe a questão da performance do sistema: podem existir centenas de clientes a fazer pedidos ao servidor a qualquer momento, o que implica que um pequeno atraso no sistema seja problemático. Para minimizar a latencia, os programadores tentam levar algumas tarefas de processamento para a máquina cliente ou para outras máquinas localizadas "perto" do servidor denominadas por "middleware".

A WWW é actualmente um enorme sistema cliente/servidor. Na realidade é um pouco mais complexo, pois temos muitos servidores e clientes a coexistir numa única rede em simultâneo.

Inicialmente era efectuado um pedido a um servidor que devolvia um ficheiro que o "browser" ("software" para visualizar páginas da WWW) sabia interpretar formatando-a na máquina local. Rapidamente, houve necessidade de não só disponibilizar páginas de um servidor, mas também ter acesso a todas as capacidades de um sistema cliente/servidor. Pretendeu-se que o cliente enviasse informação para o servidor, por exemplo que fosse possível adicionar dados no servidor. Estas são algumas das alterações que temos visto

na WWW.

Um "browser" da WWW foi um avanço relativamente ao conceito de permitir que um bloco de informação seja visualizada em qualquer tipo de computador sem alteração. Mas, os "browsers" mostraram-se primitivos, relativamente aos pedidos funcionalidades que lhes exigiam. Estes não eram particularmente interactivos e tendiam a ser lentos porque qualquer tarefa de programação teria de ser executada no servidor. Eram necessários muitos segundos ou minutos para descobrir que tínhamos cometido um erro qualquer no pedido feito ao servidor. Como o "browser" era apenas um visualizador não podia desempenhar a mais simples tarefa de computação (por outro lado, era seguro, porque não podia executar nenhum programa na máquina local que contivesse erros ou vírus). Para resolver este problema, foram feitas outras aproximações ao problema. Características gráficas foram adicionadas aos "browsers" que permitem agora a visualização de melhores animações e de vídeo. Porém, a totalidade do problema só pode ser resolvido incorporando a capacidade de executar programas no cliente, sobre um "browser".

O desenho inicial do "browser" de WWW permitia a inclusão de um conteúdo interactivo, mas a interacção era providenciada completamente pelo servidor. O servidor disponibilizava páginas estáticas ao browser, que simplesmente as interpreta e visualiza. O HTML básico contém mecanismos muito simples para a introdução de dados: campos de texto, "checkboxes", "radioboxes", listas e botões que apenas podem ser programados para fazer uma de duas acções: limpar os dados do formulário ou submete-los ao servidor. O envio dos dados passa por uma CGI ("Common Gateway Interface") que existe em todos os servidores da WWW. O texto que é enviado indica à CGI o que tem de fazer com os dados; a acção mais comum é executar um programa que está localizado no directório `cgi-bin` do servidor. Estes programas podem ser criados em qualquer linguagem, mas a escolha mais comum é o Perl porque foi desenhado para fazer manipulação de texto e é interpretado, o que permite ser instalado em vários servidores independentemente do processador e do sistema operativo.

Muitos sites da WWW são construídos apenas recorrendo às CGI, e de facto é possível fazer praticamente tudo recorrendo a estas. O problema é o tempo de resposta. O tempo de resposta de uma CGI depende da quantidade de dados que têm de ser enviados da leitura dos mesmos pelo servidor e na capacidade da Internet (para além de que o iniciar uma CGI tende a ser um processo lento). Os "desenhadores" iniciais da WWW não previram o aumento significativo da necessidade de largura de banda para as aplicações desenvolvidas posteriormente.

A solução passa por executar programas no cliente. Grande parte das máquinas que executam "browsers" da WWW são máquinas poderosas capazes de executar várias tarefas. Com o HTML estático original, as máquinas apenas ficavam à espera que o servidor lhes enviasse a próxima página. A execução de programas no cliente significa que o "browser" da WWW faz todo trabalho que pode, o que resulta num processo muito mais rápido e interactivo.

Um dos passos importantes na programação no cliente foi o desenvolvimento dos "plug-ins". Esta é uma forma de adicionar novas funcionalidades ao "browser" descarregando ("downloading") código que se liga ao browser. Cada "plug-in" indica ao "browser" que este pode fazer agora uma nova tarefa. É possível adicionar ao "browser" um comportamento rápido e poderoso via "plug-ins", mas criar um "plug-in" não é uma tarefa trivial e não é algo que se pretenda fazer com o objectivo de ser utilizado num "site" em particular. O grande valor de um "plug-in" é que permite a um programador desenvolver uma nova linguagem para a programação em clientes e adicionar essa linguagem ao "browser" sem a permissão do criador do "browser". Assim, os "plug-ins" permitem a criação de novas linguagens para clientes.

Os "plug-ins" resultaram da grande quantidade de linguagens "script". O código fonte de uma linguagem "script" é embebido na página HTML para ser executado no cliente. O "plug-in" que interpreta a linguagem é automaticamente activado enquanto a página HTML está a ser visualizada. As linguagens "script" tendem a ser razoavelmente simples, e porque os seus programas são apenas texto que faz parte da página de HTML são rapidamente carregados quando é pedida a página que o contém.

A desvantagem é o código do programa estar exposto a todos, apesar de que normalmente não se fazem aplicações muito sofisticadas neste tipo de linguagens. Esta questão realça o facto de as linguagens "script" terem sido criadas para resolver alguns tipos de problemas específicos, principalmente a criação de interfaces mais ricos e interactivos.

As linguagens de "script" mais conhecidas actualmente são o "JavaScript", o "Vbscript" (que se assemelha ao "Visual Basic") e "Tcl/Tk".

O "JavaScript" é provavelmente destas linguagens a mais suportada; esta é incluída no "Netscape Navigator" e no "Microsoft Internet Explorer", os dois "browsers" mais utilizados actualmente.

No entanto nem todos os problemas podem ser resolvidos através de uma linguagem de "script". A solução para estes problemas mais utilizada é o Java. Esta é uma

linguagem poderosa e construída para ser segura e pode ser executada em várias plataformas. Mas também tem sido sistematicamente estendida para conter novas funcionalidades e novas bibliotecas que manipulam problemas que são difíceis nas linguagens de programação tradicionais. Destes problemas salientam-se o "multi-threading", o acesso a base de dados, programação em rede e computação distribuída. O Java permite a execução de programas no cliente via "applet".

Um "applet" é um mini-programa que apenas pode ser executado sobre um "browser" da WWW (ou uma aplicação especialmente construída para tal). Um "applet" é descarregado automaticamente como parte de uma página da WWW (como um desenho que faça parte da página) e quando activado executa um programa. Os "applets" disponibilizam uma forma automática de distribuir "software" cliente a partir do servidor, na altura que o utilizador necessita deste, e não antes. Assim obtém-se a última versão do "software" sem falha, e sem necessidade de reinstalação.

Devido à forma como o Java foi desenhado, o programador necessita apenas de criar um único programa. Este funciona automaticamente em todos os computadores que tenham um "browser" com um interpretador de Java (o que grande parte dos "browsers" já possuem).

Como o Java é uma linguagem de programação bastante completa, grande parte do trabalho pode ser efectuado no cliente antes deste fazer qualquer pedido ao servidor.

Uma das vantagens que os "applets" têm sobre as linguagens de "script", é o facto de ser distribuída num formato já compilado, o que leva a que o código fonte não esteja à disposição do cliente.

3.2.1 JDBC

O acesso às bases de dados é implementado utilizando um "pacote" de objectos Java, especialmente criado para efeito - o **JDBC**.

O JDBC é um interface de acesso a bases de dados SQL. Foi desenvolvido pela JavaSoft com o objectivo de facilitar aos programadores de Java um interface uniforme a um largo espectro de base de dados relacionais ([GHRC]).

A API (biblioteca de objectos) do JDBC define classes de Java que representam ligações à base de dados, questões SQL, conjuntos de resultados, etc. Também permite ao programador Java fazer pedidos SQL e processar os resultados. O JDBC é a principal API para aceder à base de dados através do Java. Uma operação típica

consiste no estabelecimento de uma conexão à base de dados, na execução de uma consulta SQL, e no envio da resposta da base de dados.

SQL - (Structured Query Language) é a linguagem "standard" para enviar e receber informação de um base de dados. As consultas tomam a forma a de um comando que permite seleccionar, inserir, actualizar, pesquisar dados, etc. [SQL92]

A API do JDBC é implementada via um gestor de "drivers" que pode suportar múltiplos "drivers" ligando diferentes bases de dados. Os "drivers" do JDBC podem ser inteiramente escritos em Java sendo por isso possível fazer o descarregamento da Internet dos mesmos, ou podem ser implementados usando métodos nativos para comunicar com bibliotecas de acesso a base de dados já existentes. Uma prova da crescimento do JDBC é o número crescente de drivers JDBC escritos por vendedores de base de dados em Java.

A utilização do JDBC vai permitir uma maior portabilidade da aplicação relativamente ao suporte de base de dados utilizado. Desta forma, o acesso à base de dados não depende das características da máquina de onde o acesso se está a processar, visto que o "driver" que permite a comunicação com o servidor ser uma classe Java que vem juntamente com o "applet".

Com a utilização do JDBC o gestor do sistema tem como únicos pré-requisitos que a base de dados tenha drivers para JDBC, ou que suporte ODBC. Estes pré-requisitos não são considerados restritivos dado que uma grande parte das base de dados comerciais já suportam ODBC, e é esperado que drivers compatíveis com JDBC estejam disponíveis para as principais bases de dados comerciais.

3.2.2 "Serialização"

A ideia da "serialização" é escrever objectos de dados em série. Esta possibilidade, que é fundamental para uma grande número de aplicações, permite-nos escrever objectos de dados para ficheiros e para canais de comunicações. Para que este processo seja útil, tem que ser reversível. Ou seja, tem de ser possível recuperar um objecto análogo ao que inicialmente foi "serializado" - **"desserialização"**. Nesta questão é que difere a "serialização" de outras formas de saída de dados. Na "serialização" de objectos não é perdida a estrutura inicial do objecto.

A "serialização" tradicional é normalmente executada de uma forma binária (e não em formato de texto), principalmente para minimizar a quantidade de tempo despendido

na transmissão de dados ou na quantidade de espaço necessário para armazenar os dados. Estas razões pragmáticas levaram à decisão de como o formato da "serialização" deveria estar estruturado.

Com a evolução dos sistemas de comunicação de computadores, a "serialização" tornou-se comum. Um dos exemplos mais conhecidos de "serialização" é o pacote de protocolos RPC/XDR que não só contém um formato "standard" para a "serialização" de dados, o "eXternal Data Representation" ou XDR, mas também tenciona automatizar o processo de "serialização" (chamado de RPC). O RPC de facto vai mais além ao não só permitir que os dados sejam convertidos para e de um formato "serializado", mas também por permitir que os dados façam parte da execução de um processo remoto. Este facto é a razão pelo nome RPC - "Remote Procedure Call" - que é dado ao pacote como um todo. Mais recentemente outras tentativas foram feitas como o CORBA, ASN-1 e um conjunto de API's da Microsoft. A Sun também apresentou a sua solução ao criar o pacote RMI - "Remote Method Invocation".

Para se entender a simplicidade da solução de "serialização" disponibilizada pelo Java, devemos observar o que era necessária fazer anteriormente. Os sistemas tradicionais orientados por objectos utilizavam a facilidade de incluir procedimentos em instâncias de objectos de dados para permitir criar um interface de "serialização" e do processo inverso para cada objecto. Estes interfaces são responsáveis por ler de e escrever para um "stream" de dados. Em geral cada procedimento apenas trabalha com apenas um campo da estrutura de dados de um forma directa e para cada tipo de dados mais complexo, simplesmente invoca rotinas similares ao tipo de dados mais complexo. Na realidade este método funciona em qualquer linguagem, só que nas linguagens orientadas por objectos permitem-nos "empacotar" dados com rotinas que sabem como manusear os dados de um forma mais natural e que de facto leva a uma maior simplificação e facilidade na manutenção. Estes são de facto alguns dos benefícios das linguagens orientados por objectos.

Apesar da metodologia ser simples, existem algumas observações que devem ser efectuadas e algumas conclusões que podem ser retiradas.

As classes podem conter componentes de dados provenientes do processo de "serialização". Este facto pode-se dever por questões de segurança ou porque os campos são derivados de outros. Nesta último caso, o método que efectua a leitura pode escolher recalcular automaticamente o campo.

Se as classes estão contidas umas nas outras ou se existirem listas ligadas, é necessário tomar precauções quanto a ciclos infinitos, se não existir protecção contra recursão

infinita. Apesar de ser rara a transmissão destas estruturas para máquinas remotas ou o armazenamento destas em ficheiros, este problema deve ser levado em conta.

A "serialização" de uma classe em Java é feita pela implementação pela classe do interface `java.io.Serializable`. As classes que não implementarem este interface não poderão ter nenhum dos seus estados "serializado". Todos os sub-tipos de uma classe "serializavel" são também "serializaveis". O interface de "serialização" não tem métodos ou campos e serve apenas para identificar a semântica de se ser "serializavel".

Para permitir que sub-tipos de classes não "serializaveis" sejam "serializaveis", o sub-tipo pode assumir a responsabilidade de gravar e de restaurar o estado dos campos públicos e protegidos. Os sub-tipos podem assumir esta responsabilidade apenas se a classe que é estendida tem um construtor sem argumentos que inicializa o estado da classe.

No decorrer da "serialização", os campos de classes não "serializaveis" são inicializados utilizando o construtor que não aceita argumentos. Um construtor sem argumentos tem de estar acessível à subclasse que é "serializavel". Os campos de subclasses "serializaveis" serão restaurados através do "stream".

As classes que necessitem de tratamento especial no decorrer do processo de "serialização" e de "desserialização" têm que implementar os seguintes métodos:

- `writeObject` - grava um objecto para um ficheiro;
- `readObject` - lê um objecto de um ficheiro.

O método `writeObject` é responsável pela escrita do estado de um objecto para que o método `readObject` seja capaz de o reconstituir. O mecanismo utilizado por defeito na gravação dos campos do objecto pode ser chamado invocando `out.defaultWriteObject`. O estado é gravado escrevendo individualmente os campos para o `ObjectOutputStream` utilizando o método `writeObject`, ou utilizando o método `DataOutput` que é suportado pelos tipos de dados primitivos (`int`, `float`, `char`, etc.).

O método `readObject` é responsável pela leitura de um "stream" e posterior restauro dos campos da classe. Este método pode chamar `in.defaultReadObject` para invocar o mecanismo existente por defeito para restaurar os campos não estáticos do objecto. O método `defaultReadObject` utiliza informação do "stream" para atribuir os valores dos campos do objecto aos nomes do objecto corrente. Esta situação cobre os casos em que a classe evoluiu para conter novos campos. O método não necessita de se preocupar com o estado da sua superclasse ou subclasse.

Os objectivo da "serialização" de objectos em Java são:

- Ter um mecanismo simples e extensível de armazenar objectos;
- Manter o tipo de um objecto Java numa forma "serializada";
- Ser extensível para suportar a persistência de objectos Java;
- Permitir ao objecto definir o seu formato externo.

A título de exemplo apresentamos o código que executa a leitura de um ficheiro e a "desserialização" de um objecto da classe Project.

```
URL url = new URL(getCodeBase(), fileName);
ObjectInputStream in = new ObjectInputStream(url.openStream());
project = (Project)in.readObject();
```

Após estas linhas de código, o objecto `project` contém o resultado da "desserialização" do objecto armazenado no ficheiro `fileName`.

Capítulo 4

Implementação do sistema

Neste capítulo começamos por analisar a estrutura e utilização da classe BlueBase. Seguidamente é descrito o utilitário BindData que assiste na construção de um objecto BlueBase a partir de uma descrição gráfica vectorial do projecto e de uma base de dados contendo informação sobre os seus constituintes. São também analisados alguns detalhes da implementação do sistema.

4.1 Um objecto BlueBase

Após ter sido criado o objecto da classe BlueBase, este é gravado num ficheiro (com a extensão `.jxf`). O processo de gravar um objecto para um ficheiro denomina por **"serialização"**. Este objecto contém a definição dos espaços (áreas), dos caminhos e dos blocos, bem como a associação dos espaços e blocos a entradas na base de dados.

A informação existente na base de dados não é incorporada neste ficheiro, devido ao facto das bases de dados normalmente conterem grandes quantidades de informação. No entanto, o acesso à base de dados é efectuado através da existência de um servidor de base de dados. Se o servidor for acedido de forma remota, o nome da base de dados deve incluir o endereço de rede (ex.: `www.ncc.up.pt\xxx.db`). Se for acedido localmente é suficiente a introdução do nome da base de dados (ex: `xxx.db`). São também mantidas informações relativas ao protocolo e à porta utilizados para comunicar com a base de dados.

As aplicações finais após efectuarem a leitura do objecto da classe BlueBase podem invocar os métodos que necessitarem. Duas diferentes aplicações invocarão diferentes

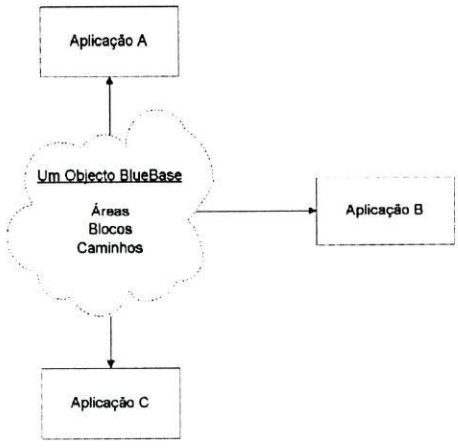


Figura 4.1: Utilização objecto da classe BlueBase.

métodos, apesar de poderem utilizar ambas o mesmo objecto (figura 4.1).

Os objectos BlueBase desempenham uma papel central nesta proposta de integração de dados gráficos e estruturados. Estes, são formados por um conjunto de outros objectos que descrevem os elementos existentes no projecto (figura 4.2).

Para além de áreas, blocos e caminhos estes objectos também contêm um conjunto de informações de carácter geral:

Nome do projecto - tem o propósito de identificar o objecto BlueBase. Dado que o nome pode não coincidir com o nome dado ao ficheiro que contém o objecto, esta propriedade deve ser utilizada para descrever sucintamente o que se pretende representar no objecto.

Referência à localização da base de dados - permite ao objecto BlueBase estabelecer a conexão com a base de dados. A localização da base de dados, é composta pelo endereço de rede, protocolo de acesso à base de dados e porta de comunicações disponibilizada pelo servidor de base de dados.

Lista de propriedades - é composta por pares de elementos da forma nome da propriedade/valor da propriedade. Esta lista deve ser utilizada para manter informação genérica como a data em que o projecto foi concebido, o autor do projecto, etc. A razão que levou à criação de listas personalizáveis de propriedades em vez de campos fixos de dados para a manutenção deste tipo de informação, foi a flexibilidade necessária dada a abrangência de tipos de aplicações que se pretende abranger.

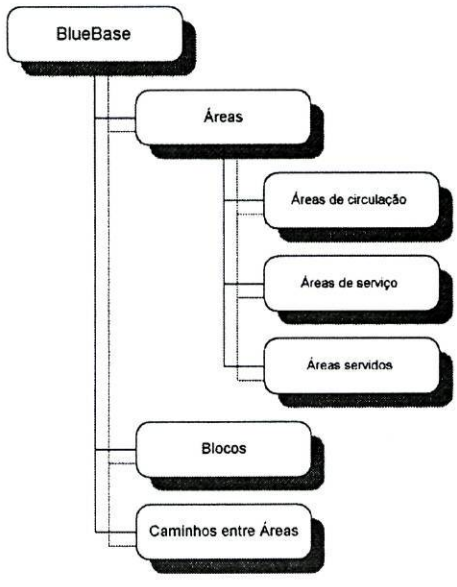


Figura 4.2: Estrutura de um objecto BlueBase.

Limites do projecto - os limites do projecto correspondem às coordenadas noroeste e sudeste da representação gráfica. As coordenadas têm como principal objectivo marcarem os extremos da representação gráfica, algo necessário para a implementação de um interface gráfico.

4.1.1 Áreas

Cada área é um objecto que define um espaço fechado. Este objecto, à imagem dos objectos BlueBase, contém informação gráfica, informação estruturada e um conjunto de métodos que nos permitem manipular cada área.

Cada área contém o seguinte conjunto de dados:

Nome da área - identifica a área a que o objecto se refere;

Vértices do polígono da área - determina a representação gráfica da área;

Tipo de área - define o tipo de espaço (circulação, serviço ou servida) que esta área define;

Propriedades - conjunto de características da área à escolha do criador do projecto;

Caminhos - conjunto de áreas com quem tem interface de comunicação.

Como ilustrado na figura 4.2, as áreas podem ser de três tipos distintos:

Áreas de circulação - espaços que existem para permitir a movimentação de pessoas entre áreas. Exemplos: corredores, átrios de entrada;

Áreas de serviço - espaços relacionados com um serviço do edifício e que não estão abertos ao público. Exemplo: cozinha de um restaurante, casa das máquinas;

Áreas servidas - espaços que são servidos pelos serviços do edifício. Exemplos: zona pública de um restaurante, uma biblioteca.

Internamente é mantido um conjunto de áreas distinto para cada tipo de área. A identificação de cada área é processada através do nome da área. Assim, não é conveniente a utilização do mesmo nome para duas áreas distintas.

A associação de uma área a um registo de uma tabela da base de dados, é estabelecida pela utilização do mesmo nome em ambas as representações. Por esta razão, também não é conveniente a existência de dois registos numa tabela de áreas com o mesmo nome de área. O nome da área deverá ser um **campo chave** (coluna de uma tabela de base de dados que é utilizada para identificar os registos da tabela. Esta é utilizada como índice no acesso à tabela. Pode ser necessário uma ou mais colunas para identificar de forma única um registo [JUJW97]).

Cada uma das áreas pode conter uma lista de outras áreas com quem comunica, determinando a existência de um caminho entre essas duas áreas. Cada entrada desta lista é composta por dois elementos: área com que comunica e a distância entre os pontos médios de cada área. A planta representada na figura 4.3 visualiza nove áreas (três corredores e seis salas). Entre os corredores adjacentes existe um ligação. O símbolo da porta também representa uma ligação entre salas e sala com corredor. As ligações possíveis estão representadas na figura 4.4.

Neste exemplo a "Sala 5" tem na lista de caminhos as seguintes entradas ("Corredor B",10) e ("Sala 6", 14), sendo os valores 10 e 14 as distâncias entre os pontos médios de "Sala 5" e "Corredor B", e entre a "Sala 5" e "Sala 6" respectivamente.

4.1.2 Blocos

Os blocos são objectos que representam equipamento existente num edifício (figura 4.5). Cada objecto bloco é composto pelo seguinte conjunto de dados:

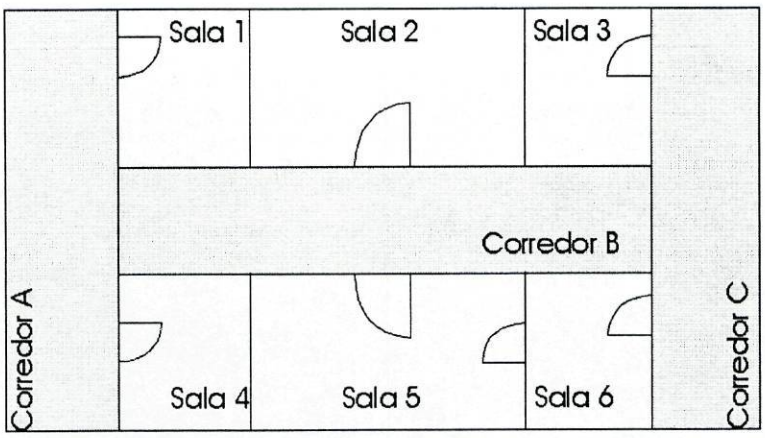


Figura 4.3: Representação de espaços e portas.

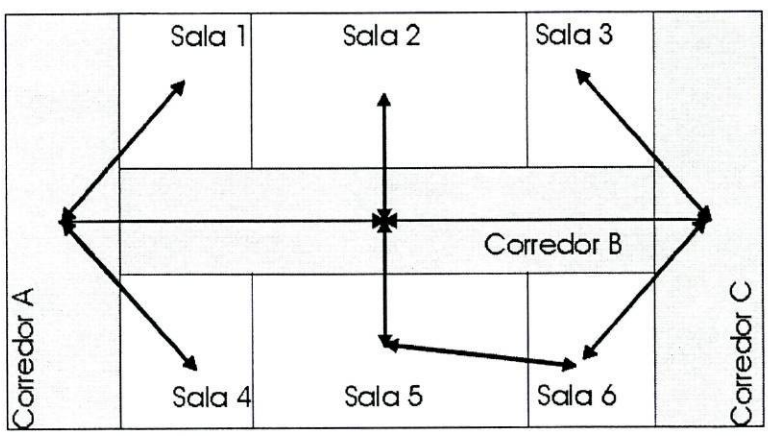


Figura 4.4: Representação dos caminhos existentes entre espaços.

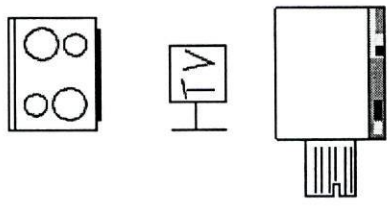


Figura 4.5: Exemplos de tipos de blocos.

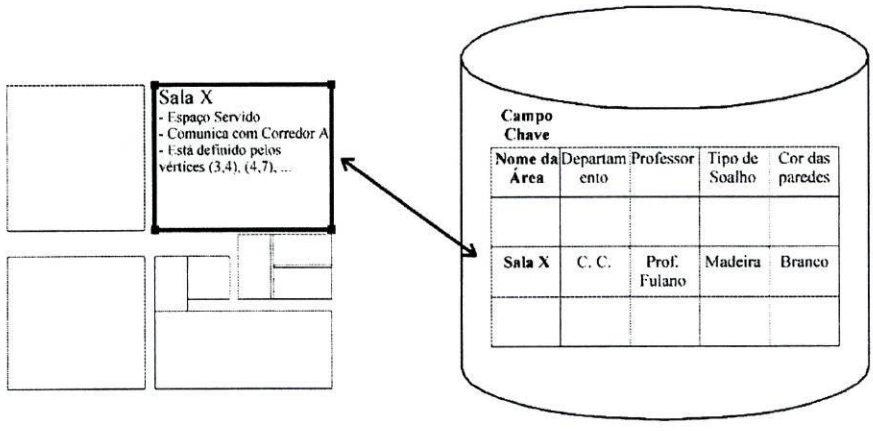


Figura 4.6: Associação de um objecto área a um registo através do campo chave.

Nome do bloco - identifica cada bloco (Ex.: "Fogão do apartamento 4", ou "Fotocopiadora C");

Nome do tipo de bloco - identifica o tipo de bloco e por consequência a sua representação gráfica (Ex.: "Fogão", "Televisão", "Fotocopiadora");

Coordenadas da origem - localização do bloco no edifício;

4.1.3 Associação de elementos à base de dados

A associação de cada área e cada bloco à base de dados é feita através do seu nome. Uma área ou um bloco encontra-se associado a um registo de uma tabela da base de dados quando o seu nome é idêntico ao valor do campo chave da tabela que descreve os elementos do seu tipo. A figura 4.6 ilustra a associação entre uma área designada por "Sala X" e um registo da tabela de áreas cujo campo chave ("Nome da Área") tem o valor "Sala X".

O nome do objecto é usado como índice para aceder ao registo que contém as propriedades do objecto.

Do mesmo modo, para identificar um objecto que esteja associado a uma entrada numa tabela, basta procurar o objecto cujo nome é igual ao valor do campo chave.

4.2 A utilização de um objecto BlueBase

Até à obtenção de um "site" que disponibiliza uma aplicação que trabalha com um objecto da classe BlueBase "serializado", são necessários os seguintes passos:

- Criação de um applet que para além de ler o objecto "serializado" disponibilize tarefas que podem ser executadas sobre o objecto;
- Criação de uma página HMTL que faça referência ao "applet";
- Disponibilizar um servidor de base de dados para o acesso à informação estruturada.

O "applet" a ser criado deverá inicialmente ler do servidor o ficheiro originado na "serialização" do objecto da classe BlueBase. Este é posteriormente convertido num objecto de forma que o applet seja capaz de trabalhar com ele.

O objecto projecto que foi criado contém a localização da base de dados tal como que protocolos que deve utilizar para aceder a esta.

Após a "desserialização" do objecto da classe BlueBase, o "applet" deverá invocar os seus métodos de forma a comunicar com a base de dados, bem como para o representar graficamente. Desta forma, o número de linhas de código necessárias para a implementação de um "applet" (ou de uma aplicação) que utilize este sistema é reduzido. As principais tarefas podem ser desempenhadas pelos métodos que compõem cada projecto.

A figura 4.7 explica o ambiente mais lógico em que o sistema pode funcionar. É de realçar porém, que não existe nenhum impedimento que o servidor de páginas de Internet seja o mesmo computador que o servidor da base de dados.

Após o acesso ao "site" que contém a página HTML criada, é descarregado no computador do utilizador o ficheiro com descrição da página HTML. Como esta página faz referência a um "applet", o "browser" pede ao servidor de páginas o ficheiro que contém o "applet", que é então executado pelo "browser". Posteriormente, o "applet" pede ao servidor que lhe envie o ficheiro que contém a "serialização" do objecto da classe

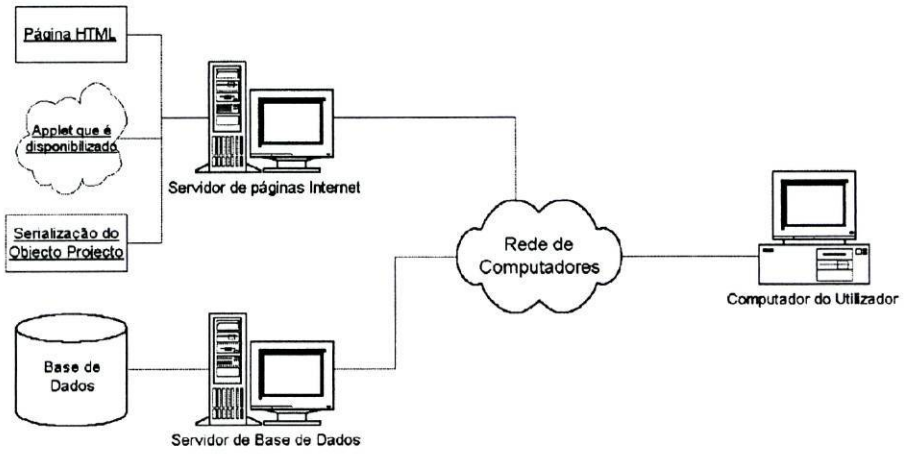


Figura 4.7: Esquema de acesso ao sistema.

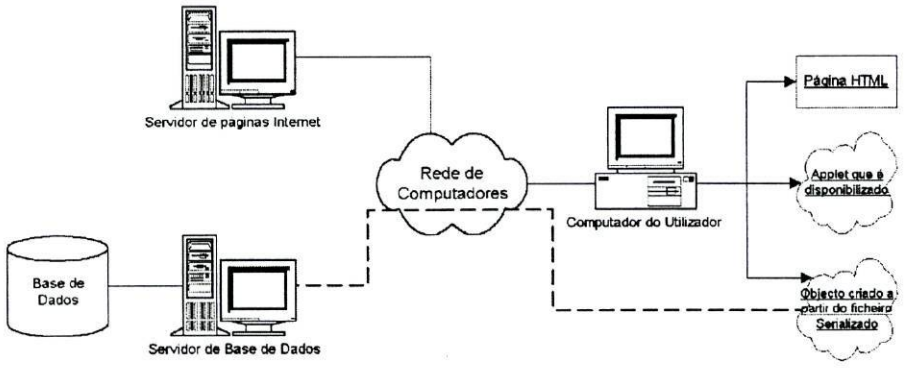


Figura 4.8: Esquema de utilização do sistema.

BlueBase. Este ficheiro é então utilizado para criar o objecto projecto na máquina do utilizador (aqui a funcionar como cliente). Quando são invocados métodos que tenham necessidade de aceder à base de dados, o objecto da classe BlueBase acede ao servidor da base de dados e gere o processamento de pedidos como explica a figura 4.8.

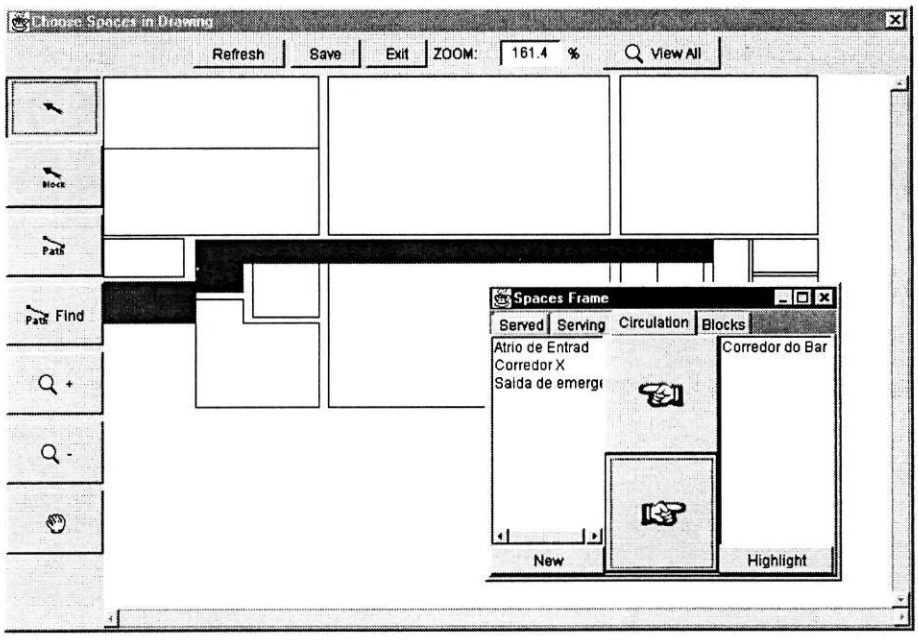


Figura 4.9: Interface gráfico do utilitário BindData.

4.3 O utilitário BindData

O utilitário BindData é uma programa que auxilia o utilizador na identificação dos espaços e blocos existentes num ficheiro de representação gráfica vectorial e que permite criar ligações com registos duma tabela duma base de dados relacional.

O processo não é automático. O interface gráfico do utilitário (figura 4.9) apresenta dois conjuntos de informação e permite de uma forma simplificada seleccionar registos e espaços (ou blocos) usando um mecanismo de apontador (ex.: rato).

A informação resultante deste processo interactivo é guardado num objecto BlueBase que é posteriormente carregado pela aplicação que faz uso da integração de dados.

4.3.1 Selecção de dados

A primeira fase é composta pela introdução de dados relacionados com questões de entradas e saídas de ficheiros. O interface criado requer a introdução dos seguintes dados:

- Nome do objecto a criar;
- Representação gráfica vectorial;

- Acesso a base de dados.

O primeiro parâmetro a introduzir é o nome do ficheiro que vai albergar o objecto final. O ficheiro só será criado após a finalização do utilitário BindData. Este ficheiro deverá ser posteriormente colocado num local acessível às aplicações finais.

De seguida, deve ser seleccionado o ficheiro que contém a representação gráfica. Este deve existir no formato DXF. Após a criação do objecto final, deixa de ser necessário para o projecto o acesso ao ficheiro DXF, dado que foi retirada deste toda a informação necessária para a visualização do projecto.

A parametrização do acesso à base de dados num ambiente distribuído tem como objectivo informar o objecto final da localização, do protocolo e da porta de comunicações a utilizar na consulta à base de dados. O nome do ficheiro poderá ser composto por um endereço de rede (URL), no caso da base de dados ser acedida remotamente.

Com o objectivo de facilitar o processo de criação do objecto, é possível utilizar uma localização diferente da base de dados durante a execução do utilitário BindData. Desta forma, não é necessário existir um acesso ao sistema de distribuição final nesta fase de criação do objecto.

É executada a selecção de tabela para cada tipo de elemento do projecto. É possível seleccionar diferentes tabelas para áreas servidas, áreas de serviço, áreas de circulação e blocos. Após a selecção de tabelas, devem ser escolhidos os campos destas tabelas a serem utilizados na associação. Estes campos devem ser campos chave. A associação entre os elementos do projecto e os registos das tabelas da base de dados tem como base a utilização do mesmo nome em ambas as representações.

Não existe nenhum impedimento na selecção da mesma tabela para fazer associações com elementos diferentes. A base de dados pode conter apenas uma tabela para todos os tipos de áreas e esta ser utilizada sem alterações no sistema.

4.3.2 Processamento gráfico

A informação contida no ficheiro da representação gráfica, tem de ser processada de forma a ser utilizada pelo utilitário BindData. O processamento desta informação está dividido em duas fases: escolha de camadas que contém os dados pretendidos e linearização dos dados gráficos.

Selecção das camadas do DXF Os programas de desenho assistido por computa-

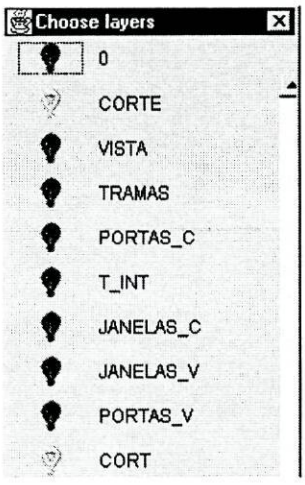


Figura 4.10: Interface para escolha de camadas do DXF.

dor habitualmente permitem dividir a representação gráfica em múltiplas camadas. Normalmente a cada camada está associado um tipo de elementos no desenho. Esta característica é muito importante para um eficiente reconhecimento de áreas fechadas, dado que nem todas as camadas contêm informação importante para esta tarefa. A existência de linhas desnecessárias pode ser um factor prejudicial à detecção dos limites de cada espaço.

O utilizador do utilitário BindData pode seleccionar as camadas ("layers") contendo os elementos para a detecção de áreas e de blocos, como está representado na figura 4.10. Depois, é executada a leitura das figuras geométricas que compõem as camadas seleccionadas.

Conversão de elementos em linhas Por questões de optimização e de facilidade de manuseamento pelo sistema, todos os arcos e elipses existentes nas camadas seleccionadas da representação gráfica original são convertidos em segmentos de recta. Esta conversão facilita a detecção de áreas fechadas, aumenta a rapidez da representação gráfica no interface e permite o cálculo mais rápido de características como áreas, perímetros e pontos médios.

Para que a conversão seja adaptada às necessidades de cada projecto o utilizador tem a possibilidade de seleccionar a forma como cada curva deve ser convertida em vários segmentos de linha. É possível introduzir o número de vértices que deverá ter o polígono que substituí cada elipse. Quanto maior for o número, maior é a exactidão no cálculo e na representação gráfica. Por outro lado, um número excessivamente alto implica uma perda significativa de performance de cálculo, bem como um aumento no tamanho final do ficheiro devido à grande

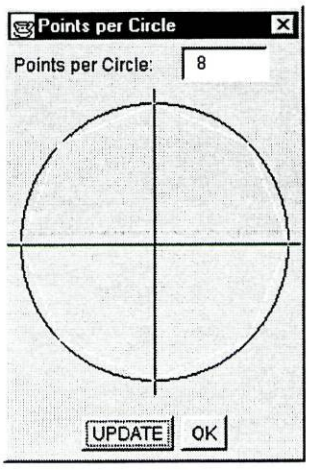


Figura 4.11: Interface para escolha do número de vértice da linearização de um círculo.

quantidade de valores a manter. As amplitudes dos ângulos serão levadas em consideração na transformação.

A figura 4.11 apresenta uma caixa de diálogo para a escolha do número de lados de um polígono regular que aproxima um circunferência. Neste exemplo cada elipse é dividida em oito linhas. A linha branca representa as oito linhas criadas, enquanto que a linha preta que se encontra sob a branca determina a elipse original. Um arco com cento e oitenta graus de ângulo é dividido em quatro diferentes linhas, ou seja, existe uma relação directa entre a amplitude do arco e o número de subdivisões desse mesmo arco.

4.3.3 Identificação das áreas

Uma das tarefas mais importantes na elaboração de um projecto é a identificação dos espaços existentes. Esta tarefa utiliza as linhas provenientes das transformações descritas anteriormente.

A identificação das áreas é processada interactivamente entre o utilizador e a aplicação. Após ter sido identificado um ponto que se encontra no interior do espaço, o programa tenta delimitar o espaço fechado que o contém. É então invocado um método que tendo como parâmetros de entrada as coordenadas do ponto dado e um conjunto de segmentos de linha, encontra o polígono que contém o ponto.

Para encontrar os vértices do polígono fechado, o método utiliza um algoritmo geométrico. Este percorre os vértices sempre no mesmo sentido até encontrar um vértice já visitado (figura 4.12).

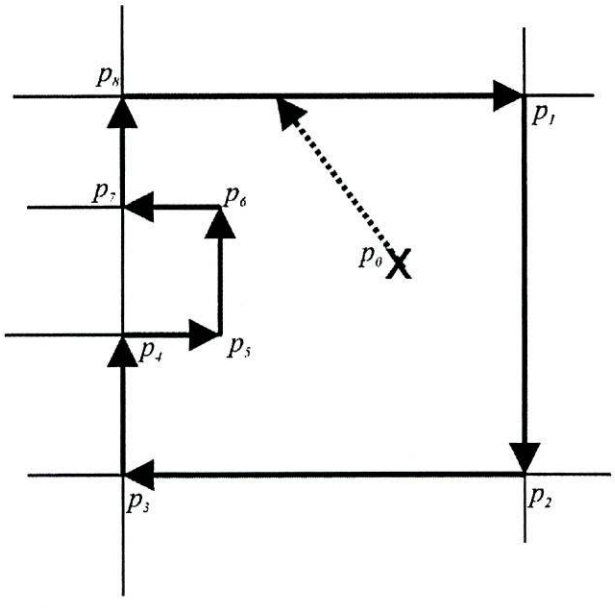


Figura 4.12: Execução do algoritmo geométrico.

```
Seja p0 o ponto inicial.
Sejam l1, ..., ln os segmentos de recta existentes no projecto.
Calculam-se p1, ..., pi vértices do polígono encontrado

r0 = primeira_linha(p0);
p  = primeiro_vertice(r0, [l1, ..., ln]);
i  = 0;
fazer
        i  = i + 1;
        pi = p;
        r1 = encontra_linha(p, r0, [l1, ..., ln]);
        p  = intersecção(ro, r1);
        r0 = r1;
até que exista um pj com 1 <= j <= i tal que pj = p
```

Quando este processo termina $[p_1, ..., p_i]$ define os vértices do polígono encontrado. Este, é utilizado na criação de um novo objecto que define o espaço. De seguida, é descrito o funcionamento dos vários métodos anteriormente referidos.

Encontrar primeira linha - `primeira_linha`

A primeira tarefa do algoritmo é encontrar um segmento de recta (parede) que faça parte do limite do espaço a ser reconhecido. Para encontrar esse segmento,

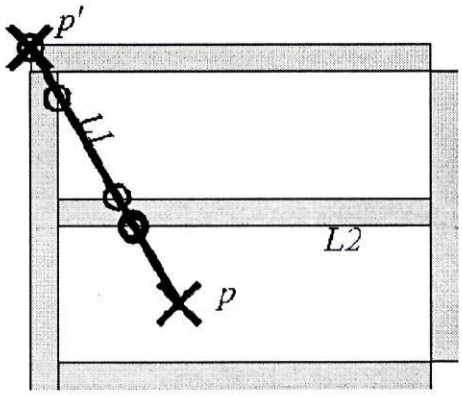


Figura 4.13: Encontra a primeira linha.

é inicialmente criado outro segmento de recta com origem nas coordenadas do ponto que inicialmente foi dado pelo utilizador e destino no limite superior esquerdo (noroeste) do desenho, como mostra a figura 4.13.

O ponto indicado pela letra p, é o ponto escolhido inicialmente pelo utilizador. O ponto p' representa o canto superior esquerdo do projecto. As coordenadas deste ponto foram calculadas na leitura inicial do ficheiro DXF. O segmento de linha L1 tem como extremos os pontos p e p'.

Tendo como base o segmento L1 são encontradas as intersecções de L1 com as restantes linhas que compõem a figura. Destas é seleccionada a intersecção que se encontra mais perto do ponto p.

No caso aqui representado, são consideradas as intersecções marcadas com o círculo. Das quatro intersecções encontradas, é seleccionada a intersecção que se encontra à menor distância de p. É então memorizada o segmento de linha L2, dado ter sido este o responsável pela intersecção seleccionada. O segmento L2 representa a primeira parede encontrada do espaço.

Encontrar o primeiro vértice - `primeiro_vertice`

A estratégia utilizada para encontrar os vértices, baseia-se em percorrer as intersecções sempre no mesmo sentido. Como mostra a figura 4.14, foi escolhido o sentido contrário ao percorrido pelos ponteiros de relógio, na implementação do algoritmo.

O segmento de recta L2 é dividido em dois segmentos de recta S1 e S2. A divisão é efectuada a partir do ponto de intersecção de L1 com L2.

São então calculados os ângulos que S1 e S2 fazem com L1, $\alpha 1$ e $\alpha 2$ respectivamente (figura 4.15).

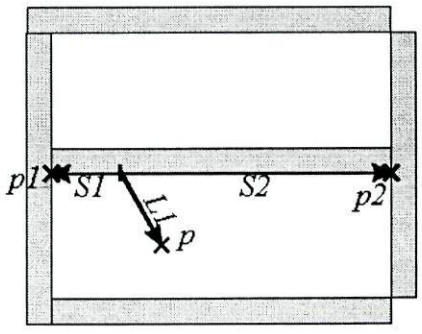


Figura 4.14: Encontra o primeiro vértice.

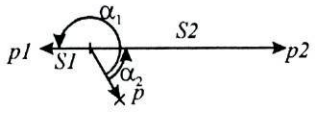


Figura 4.15: Cálculo de ângulos para encontrar o primeiro vértice.

Para encontrar o polígono que se pretende é necessário que o algoritmo percorra os vértices sempre no mesmo sentido. Assim, é necessário escolher o menor ângulo ou o maior ângulo que a última "parede" faz com cada uma das novas "paredes" de acordo com uma orientação predefinida.

Na implementação deste algoritmo, é considerada a orientação contrária à dos ponteiros de um relógio. E são consideradas os segmentos que fazem ângulos menores com a última "parede" (último segmento).

Desta forma, o menor ângulo define qual dos segmentos S1 ou S2 deve ser considerado. O passo seguinte é encontrar a primeira intersecção que S1 ou S2 (consoante o segmento seleccionado) tem com outra linha. No caso exposto na figura 4.15, S2 seria o segmento seleccionado e consequentemente seria seleccionado p2 como o primeiro vértice do espaço que contêm p.

Encontrar nova linha - `encontra_linha`

Após ter sido encontrado o primeiro vértice, o segmento S2 toma o lugar do segmento L1, e o processo repete-se (figura 4.16).

Sempre que é necessário encontrar o menor ângulo são gerados vectores à custa dos segmentos. É de realçar que a orientação dos vectores considerados depende da função que cada segmento desempenha. Podemos ver que inicialmente é criado um vector através do segmento S2 que tem orientação contrária a um vector criado posteriormente através do mesmo segmento.

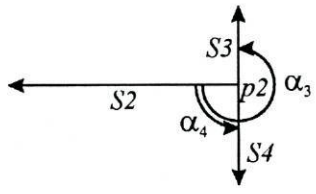


Figura 4.16: Cálculo de ângulos para encontrar novo vértice.

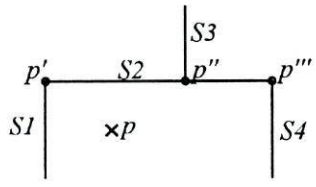


Figura 4.17: Segmentos que não "cortam" o limite do espaço.

Apenas são consideradas intersecções que cortam o segmento de recta do lado correcto, ou seja, há que analisar a intersecção para que não sejam considerados segmentos que não podem ser limites do espaço a encontrar.

Na figura 4.17 podemos constatar que existe uma intersecção entre o segmento S2 e o segmento S3. No entanto, esta intersecção não tem nenhuma influência na área fechada que se pretende determinar. Após ser encontrado o ponto p" são efectuados cálculos que demonstram se o segmento S3 corta o segmento S2 do lado pretendido. No caso de não interessar o ponto é ignorado e procura-se a seguinte intersecção (neste caso p"').

Por vezes existem situações em que segmentos de recta cruzam os espaços sem que estejam a definir um limite. Estas linhas são ignoradas, dado serem considerados erros ou linhas de orientação, e que por essa razão não devem ser consideradas. Na figura 4.18 podemos ver representados dois segmentos que dão origem às intersecções r' e r". Quando o algoritmo encontra uma extremidade como r", faz o percurso inverso até encontrar uma nova intersecção. A nova intersecção é encontrada em p"'. Os pontos que foram percorridos duas vezes (com a excepção do primeiro - p") são retirados da lista de vértices.

Quando for encontrado pela segunda vez um vértice na lista de vértices já

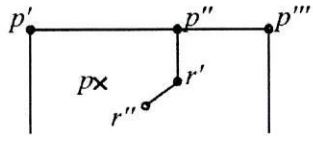


Figura 4.18: Segmentos desnessários à detecção do limite do espaço.

encontrados o processo termina. O conjunto dos vértices encontrados até esse momento define o polígono e consequentemente a área do espaço que contém o ponto seleccionado inicialmente - p.

Encontrar intersecção - O método que se encarrega de encontrar novos vértices utiliza uma função para calcular se existe alguma intersecção entre dois segmentos de recta [MP].

A função que calcula o ponto de intersecção entre dois segmentos actua da seguinte forma:

Sejam A, B, C, D dois vectores em R^2. Então os segmentos AB e CD são dados por:

$$AB = A + r(B - A), r \in [0, 1]$$
$$CD = C + s(D - C), s \in [0, 1]$$

Se existe um intersecção entre AB e CD, então

$$A + r(B - A) = C + s(D - C)$$

ou

$$XA + r(XB - XA) = XC + s(XD - XC)$$
$$YA + r(YB - YA) = YC + s(YD - YC), r, s \in [0, 1]$$

Resolvendo em ordem a r e s obtemos:

$$r = \frac{(YA - YC)(XD - XC) - (XA - XC)(YD - YC)}{(XB - XA)(YD - YC) - (YB - YA)(XD - XC)}, eq.1$$

$$s = \frac{(YA - YC)(XB - XA) - (XA - XC)(YB - YA)}{(XB - XA)(YD - YC) - (YB - YA)(XD - XC)}, eq.2$$

Seja I a intersecção, então:

$$I = A + r(B - A)$$

ou

$$XI = XA + r(XB - XA)$$

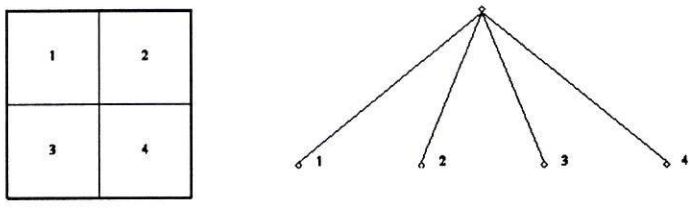


Figura 4.19: Árvore simples da "quadtree".

$$YI = YA + r(YB - YA)$$

Pela análise dos valores r e s, podem-se determinar as seguintes condições:

Se $0 <= r <= 1$ e $0 <= s <= 1$, então existe intersecção, senão não existe intersecção.

Se o denominador da eq.1 é zero, então AB e CD são paralelos. Se o numerador da eq.1 é também zero, então AB e CD são coolineares. São então calculadas as coordenadas de I se existir uma intersecção.

4.3.4 Localização de linhas

Para que este processo de identificação de intersecções não seja moroso, é necessário fazer uma pesquisa eficiente dos segmentos de linha que podem intersectar o segmento que está a ser considerado. O conjunto de segmentos de recta que podem intersectar o segmento considerado é composto por todos os segmentos existentes com a excepção dele próprio. Se não existisse nenhuma forma de estruturar os segmentos, a quantidade de segmentos considerados seria impeditivo de uma boa performance em desenhos mais complexos.

Com o objectivo de diminuir o universo de segmentos a ser considerado, foi utilizada uma forma de estruturar a representação gráfica inspirada na estruturação de dados espaciais denominada por "quadtree" [HS90] (figura 4.20). Na realidade foi efectuada uma alteração a este sistema. O sistema "quadtree" original divide a representação gráfica em quatro quadrantes não necessariamente de dimensões idênticas (figura 4.19). Cada um dos quais é sucessivamente dividido em quatro quadrantes até que cada quadrante contenha apenas um número mínimo de pontos. Este sistema é muito utilizado em representações "raster", que são compostas totalmente por pontos de várias cores, para localizar de forma eficiente as diferenças entre duas imagens semelhantes.

No entanto, este sistema tem se ser adaptado para permitir a utilização de represen-

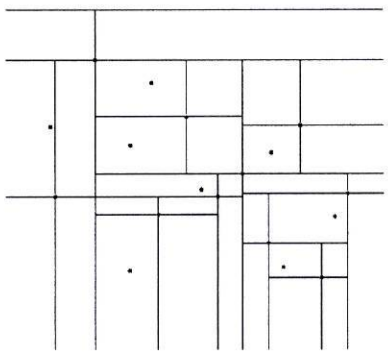


Figura 4.20: Imagem sucessivamente dividida pela "quadtree".

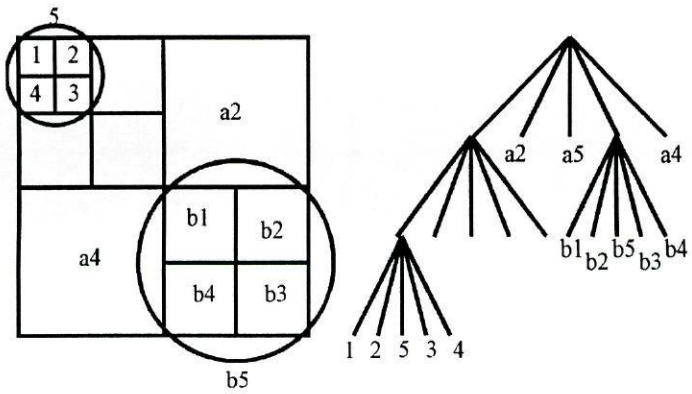


Figura 4.21: Exemplo da "quadtree" utilizada.

tações vectoriais. Enquanto que os pontos são facilmente atribuídos aos quadrantes, os segmentos de recta nem sempre podem ser atribuídos a quadrantes já que podem cruzar mais do que um quadrante. Para além dos quatro quadrantes, um quinto é composto pelos segmentos que cruzam mais do que um quadrante (figura 4.21).

Sempre que é necessário verificar que intersecções existem com um dado segmento S, são seleccionados todos os ramos da árvore que cruzam o quadrante do segmento S. No caso do segmento S cruzar vários quadrantes, é necessário seleccionar os segmentos de todos os ramos.

A divisão dos segmentos pelos cinco ramos é levada a cabo na leitura do ficheiro DXF. Os quadrantes têm no sistema adoptado tamanhos idênticos.

4.3.5 Associação de áreas à base de dados

Após ter sido encontrado o espaço fechado que contém o ponto inicial, este é visualizado no interface gráfico como demostra a figura 4.22. A área a cinzento identifica o polígono

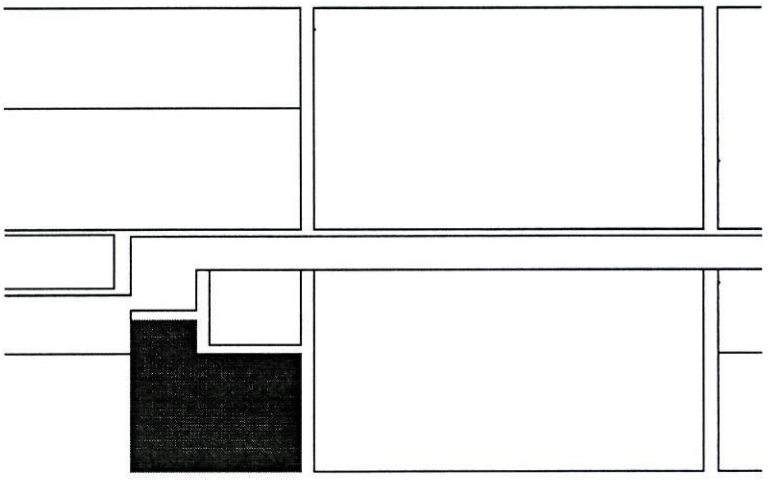


Figura 4.22: Espaço fechado encontrado.

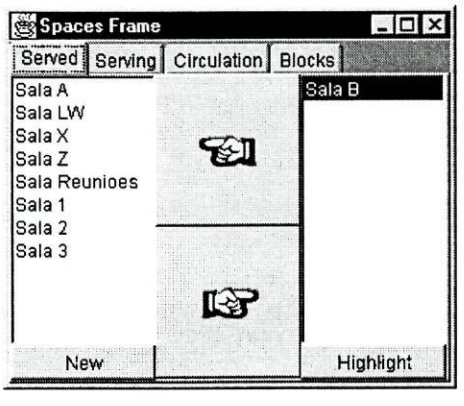


Figura 4.23: Selecção de espaços existentes na base de dados.

que representa o espaço fechado. Se a área não for a pretendida pode-se seleccionar outro ponto na tentativa de encontrar o espaço correcto.

Para que a elaboração de um objecto que define um espaço esteja concluída, é necessária a associação de um nome ao espaço. Para tal, são listados os valores residentes nas tabelas da base de dados que foram seleccionadas anteriormente como sendo as que contêm os nomes dos três tipos de espaços e dos blocos.

A associação de um nome ao polígono encontrado na representação gráfica é executada após a selecção de um nome na lista da esquerda e posterior clicar no botão que aponta para a direita. Nesta altura, é criado um objecto do tipo área, caracterizado pelo polígono seleccionado, do tipo do separador que estiver activo ("Served" - áreas servidas, "Serving" - áreas de serviço ou "Circulation" - áreas de circulação) com o nome seleccionado na lista de nomes ainda não associados (a lista da esquerda na figura 4.23). Este objecto é adicionado a uma lista armazenada numa tabela de partição

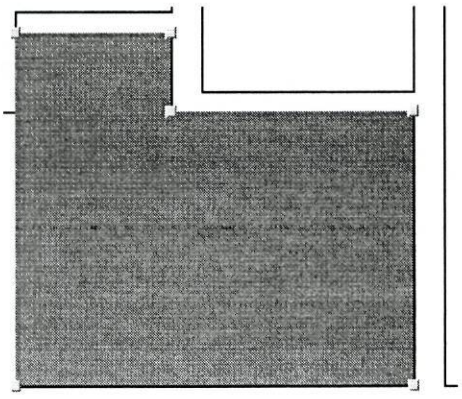


Figura 4.24: Exemplo de um espaço destacado.

relativa ao tipo de espaço do objecto. Ou seja, após a associação da área a um nome de uma tabela da base de dados, o objecto resultante é adicionado a uma lista que contém espaços de circulação, de serviço ou servidos consoante o tipo de espaço, sendo indexado pelo nome. Assim, quando for necessário obter o objecto que descreve um determinado espaço, é suficiente saber o nome e o tipo de espaço.

Para que sejam confirmados os vértices de um dado espaço, pode ser destacado um dado espaço ("highlight"). Nessa altura, cada vértice do espaço destacado, é realçado através de um pequeno quadrado cinzento (figura 4.24).

Se for necessário desfazer a associação efectuada entre determinado polígono e um nome, pode ser invocado um método de desassociação que é activado ao clicar na tecla que aponta para a esquerda.

4.3.6 Associação de blocos à base de dados

A associação de blocos processa-se de modo idêntico à associação de áreas. É necessário inicialmente seleccionar o bloco pretendido. A selecção é efectuada após seleccionar a ferramenta "Select Block" e clicar no interior do rectângulo mais pequeno que contém o bloco pretendido (figura 4.25). A selecção é confirmada mudando a cor do bloco.

Um dos separadores que contêm as listas dos elementos existentes nas tabelas da base de dados, é dedicado aos blocos ("Blocks"). A associação é concluída ao seleccionar uma entrada da lista de blocos. A selecção de um bloco associado ou a eliminação de um bloco já associado é feita da mesma forma de uma área.

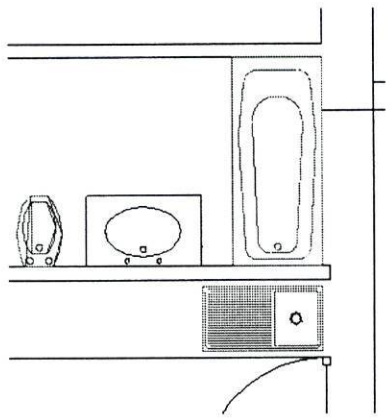


Figura 4.25: Blocos na planta.

4.3.7 Criação de caminhos entre áreas

Após terem sido criados os objectos que definem as áreas, é possível criar caminhos entre os mesmos. Para tal basta ter seleccionada uma área, ter activado o botão que diz "Path", e clicar em cima da área com que queremos que exista um caminho. Como está demonstrado na figura 4.26 é possível existirem vários caminhos a partir de uma única área. Neste exemplo, o espaço seleccionado tem ligações a três outros espaços.

Quando é criado um caminho é adicionada um par do tipo nome da área/distância na área de origem e na área de destino. Desta forma, cada objecto que descreve um espaço sabe todos caminhos disponíveis a partir deste, bem como as respectivas distâncias para os percorrer. As distâncias são calculadas entre os pontos médios de cada polígono que define cada espaço. Para que efectue a eliminação de um caminho basta efectuar o mesmo processo da criação de um caminho. Se não existir um caminho entre os espaços então é criado um novo caminho, se já existir tal caminho, então este é eliminado.

4.3.8 Gravar o objecto criado

A gravação do objecto da classe BlueBase é efectuada quando é pressionado o botão para sair do utilitário BindData. É então criado um ficheiro com o nome introduzido na primeira fase do programa.

Internamente a gravação é feita invocando um método da linguagem Java que cria um ficheiro que serve de repositório do objecto. Como todo o projecto, incluindo áreas, blocos, caminhos, etc., está incluído num objecto da classe BlueBase, basta enviar este

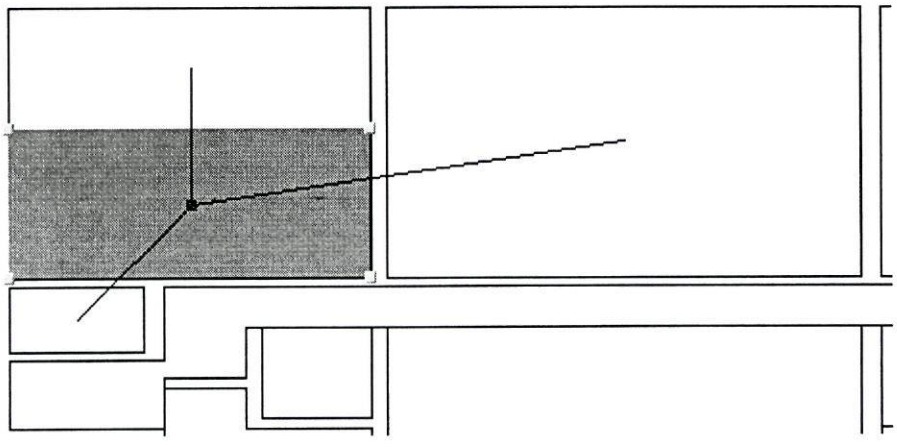


Figura 4.26: Criação de caminhos.

objecto para um ficheiro que todos os restantes são também enviados para o ficheiro (é efectuada a "serialização" do objecto).

4.4 Detalhes de implementação

4.4.1 Tabelas de partição

As tabelas de partição são implementadas nas listas que armazenam as áreas e os blocos no objecto projecto. A escolha de tabelas de partição contribuí para o aumento na performance de todo o sistema.

As tabelas de partição são estruturas para o armazenamento de dados. Estas estruturas fazem o mapeamento de valores que funcionam como índices de acesso a outros dados. As tabelas de partição comportam uma lista de elementos, sendo a sua principal característica a utilização de uma função de "hashing" na localização dos elementos. Esta função utiliza os índices atribuídos aos dados armazenados para a rápida localização dos dados na estrutura. A implementação destas estruturas na linguagem Java permite que sejam armazenados e utilizados como índice qualquer tipo de objectos.

O estudo que se segue tenciona demonstrar como estas estruturas são implementadas em Java e que outras formas concorrentes existem.

Uma das estruturas de armazenamento de dados mais comuns em Java é um vector. Cada vector permite seleccionar de uma sequência de objectos utilizando um número, ou seja associa um número a cada objecto.

No entanto, existem estruturas que permitem seleccionar de uma sequência de objectos utilizando outro critério. Uma pilha é um exemplo: o critério de selecção é "o último objecto a entrar na pilha".

Uma transformação desta ideia "seleccionar de uma sequência" é a criação de um mapeamento, um dicionário ou um vector associativo.

Conceptualmente, é idêntico a um vector, mas na substituição de um número como índice, a procura é efectuada utilizando outro objecto.

O conceito é utilizado conceptualmente em Java pela classe `Dictionary`. O interface desta classe é simples: `size()` retorna o número de elementos que contém, `isEmpty()` é verdadeiro se não existirem elementos, `put(Object chave, Object valor)` adiciona um valor (o que queremos guardar) e associa-o a uma chave (o índice para aceder ao valor), `get(Object chave)` retorna o valor correspondente ao índice chave, e `remove(Object chave)` elimina da estrutura o par chave-valor. Existem enumerações:

`keys()` produz uma enumeração de chaves, e `elements()` produz uma enumeração de valores. Estes são os métodos que compõem um `Dictionary`.

Não é difícil a implementação de um `Dictionary`. Existe uma aproximação ao problema simples, que utiliza dois vectores, um para as chaves e outro para os valores.

A primeira questão é a definição de vector associativo (`AssocArray`) que em Java é uma extensão de um `Dictionary`. O que significa que `AssocArray` é um tipo de `Dictionary`, logo podemos fazer o mesmo tipo de pedidos que se podem fazer a um `Dictionary`.

O vector de chaves e de valores estão associados através do número de índice dos respectivos vectores. Ou seja, se invocar `put()` com a chave "telhado" e o valor "azul" (assumindo que estou a associar as várias partes de uma casa às cores com que estão pintadas) e já existem cem elementos na `AssocArray`, então "telhado" será o elementos cento e um das chaves e "azul" será o elementos cento e um dos valores. Quando for invocado `get()`, e for passado "telhado" como chave, será produzido um número de índice com `chaves.indexOf()`, que é utilizado para seleccionar o valor correcto no vector de valores.

A biblioteca base do Java contém apenas uma extensão de `Dictionary`, que é a `Hashtable` (tabela de partição). As tabelas de partição do Java têm basicamente o mesmo interface que a `AssocArray`, mas diferem numa característica importante - a eficiência. Se pensarmos no que tem de ser efectuado para executar o método `get()`, parece ser bastante lenta a procura no vector por uma chave. É nesta acção que as tabelas de partição aumentam a velocidade: em substituição à procura linear por uma chave, é utilizado um código de indexação - `hashCode()` - diferente para cada objecto e utiliza-o na procura da chave. Esta característica resulta num melhoramento significativo de performance.

4.4.2 O formato DXF

Os dados da representação gráfica são importados de desenhos gravados em formato DXF ("Drawing Exchange Format").

Os ficheiros DXF são ficheiros em formato de texto ASCII. Estes são suportados por grande parte das aplicações CAD, pelo que podem ser facilmente traduzidos noutros formatos, ou submetidos a outros programas para uma análise específica.

Um ficheiro DXF é simplesmente um ficheiro de texto ASCII com extensão `".dxf"` e

que contém texto com uma formatação especial. A organização geral de um ficheiro DXF é a seguinte:

- A secção HEADER - esta contém informação geral sobre o desenho. Cada parâmetro tem o nome da variável e um valor associado;
- A secção TABLES - Esta secção contém as definições de alguns itens;
  - Tabela de tipos de linhas (LTYPE);
  - Tabela de níveis ou camadas existentes (LAYER);
  - Tabela de estilos de texto (STYLE);
  - Tabela VIEW;
  - Tabela de sistemas de coordenadas do utilizador (UCS);
  - Tabela de configuração de Viewports;
  - Tabela de gestão do desenho (DWGMGR).
- A secção BLOCKS - Esta secção contém a definição dos tipos de blocos que se podem encontrar no desenho;
- A secção ENTITIES - Esta secção contém todas as entidades do desenho, incluindo referências a blocos;
- END OF FILE - Marca o fim do ficheiro DXF.

Capítulo 5

Aplicações do Modelo Construído

O sistema proposto pode ser utilizado quer por aplicações Java com todas as funcionalidade que qualquer aplicação pode ter, quer por applets que são executados pelos "visualizadores" da WWW.

Com a seguinte descrição de diferentes aplicações, pretendemos demonstrar a flexibilidade da biblioteca disponibilizada. Os casos aqui estudados são apenas alguns que entendemos serem representativos, obviamente outros seriam possíveis.

As descrições consideradas dividem-se quanto aos elementos que descrevem. Cada área pode representar um(a):

- Espaço interior a um edifício;
- Edifício de uma área urbana;
- Sub-região de uma grande região.

Para cada uma destas representações, são apresentadas algumas aplicações que trabalham o sistema com um objectivo específico.

As consultas feitas aos dados disponibilizados nas várias aplicações dividem-se em três tipos:

- Consultas à base de dados que resultam numa representação gráfica;
- Consultas à base de dados que resultam na listagem de registos;

- Consultas a objectos gráficos que resultam na listagem de propriedades dos objectos.

São dados exemplos destes três tipos de questões nas várias aplicações escritas.

5.1 Espaço interior de um edifício

Neste tipo de descrição cada projecto descreve um edifício. As áreas definem os espaços existentes no interior do edifício. O foco das aplicações tendem a ser os espaços interiores, ou seja, salas, corredores, etc. Os blocos são utilizados na manutenção de informação relativa aos elementos existentes no edifício, como equipamento de escritório, louça sanitária, etc.

5.1.1 Navegar na planta de um edifício

Uma das possíveis utilizações deste sistema é uma aplicação que permita ao utilizador navegar no interior de um edifício.

O interface da aplicação apresenta um mapa que representa a planta do edifício. Neste mapa estão definidas as salas e as áreas de circulação como corredores que compõem o edifício. Porque foram definidos os caminhos é possível visualizar a existência de entradas e saídas de cada espaço.

Relativamente a cada espaço é mantida o seu nome, o departamento a que pertence, os seus utilizadores habituais, a extensão do telefone que existe e se esse espaço está aberto ao público.

Assim podem ser executadas pesquisas tendo como base a informação mantida sobre os espaços. (figura 5.1)

Resposta sob a forma de representação gráfica - As perguntas seguintes resultariam no destacar dos espaços que cumpram as condições.

- Onde está a Sala X ?
- Onde está a sala do Prof. Fulano ?
- Qual é a sala com extensão telefónica 34 ?

- Que salas estão abertas ao público ?
- Que espaços pertencem ao departamento de Informática ?

Resposta sob a forma de listagem de registos - Neste caso a representação é feita à imagem das obtida em base de dados tradicionais, dada a impossibilidade de as representar graficamente no mapa.

- Quais dos utilizadores da Sala X ?
- Qual a extensão telefónica da Sala Y ?

Perguntas através de objectos gráficos - Após ter seleccionado um qualquer espaço no mapa, posso perguntar a esse espaço:

- A que departamento este espaço ?
- Qual é o horário de funcionamento desta sala ?
- Qual o caminho mais perto entre a Sala X e a Sala Y ?

Estas aplicações poderiam ser usadas como sistema de orientação em edifícios complexos. Um cenário possível de utilização, seria a criação de um quiosque que permite-se aos alunos fazer perguntas a uma planta da escola.

5.1.2 Gerir um edifício

A gestão de edifícios é um tipo de aplicação com algumas semelhanças à anterior. Estas aplicações também utilizam como suporte gráfico a planta de um edifício. Para além de dados relativos às áreas, blocos e caminhos (como na aplicação de navegação) são mantidos dados necessários à manutenção de equipamentos como datas de validade e/ou garantia de equipamento, data de compra, nome do fornecedor, etc. Pode também deve ser mantida informação relativa a características dos espaços que necessitem de manutenção, soalhos, paredes, iluminação e outros.

Com estes dados podem-se fazer as seguintes consultas:

Resposta sob a forma de representação gráfica - As perguntas seguintes resultariam no destacar de blocos e de espaços que cumpram as condições.

- Destacar os extintores cuja data e validade seja anterior à data actual;

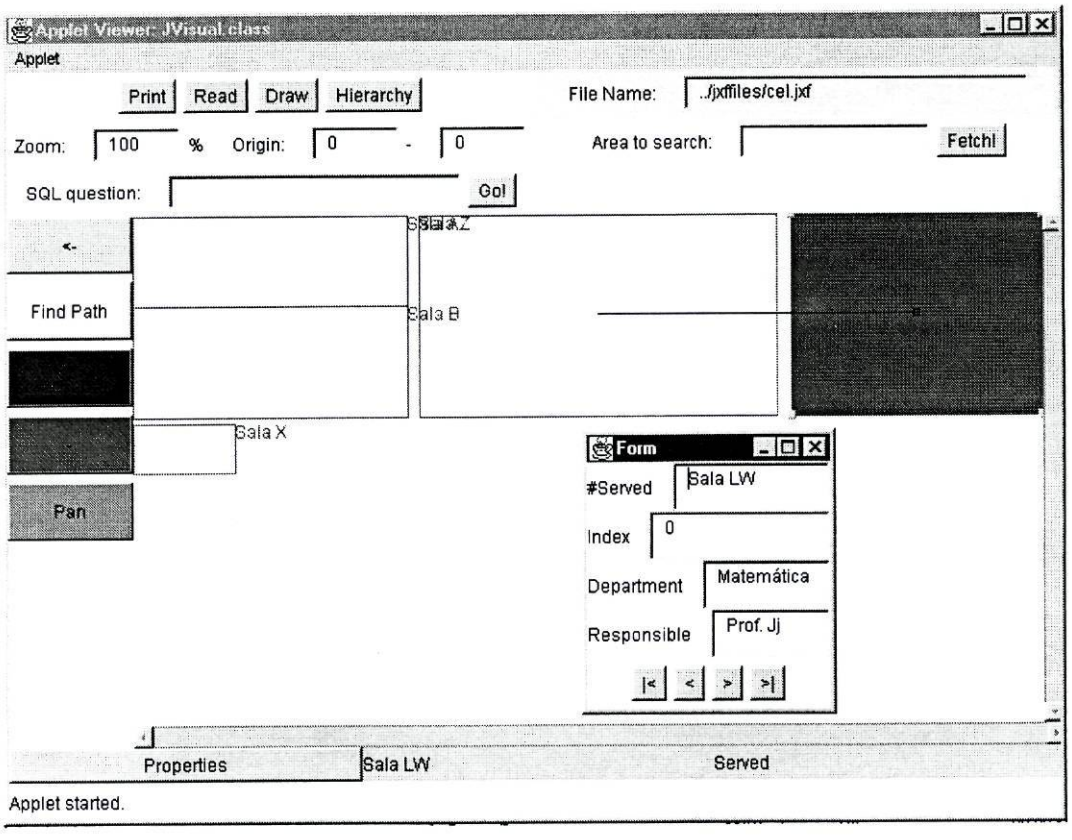


Figura 5.1: Interface do "applet" de navegação em plantas.

- Visualizar todo o material que necessita de manutenção diária;
- Destacar as salas que têm soalho de madeira.

Resposta sob a forma de listagem de registos - A resposta é visualizada em formato de texto.

- Calcular o número de aparelhos de ar condicionado existentes num determinado departamento
- Calcular a área total de um dado departamento
- Calcular a área total de espaços com soalho de madeira

Perguntas através de objectos gráficos - Após ter seleccionado um qualquer bloco ou espaço no mapa, posso perguntar a esse bloco/espaço:

- Qual é a empresa fornecedora do bloco seleccionado ?
- Qual a cor das paredes do espaço seleccionado ?

A utilização desta aplicação por um gestor de edifícios, permite-lhe a pesquisa dos vários elementos que necessitam de manutenção que existem no edifício.

5.1.3 Avaliar um projecto

Esta aplicação tem objectivos muito diferentes das anteriores. Esta, visa automatizar o processo de avaliação de um processo em termos arquitectónicos.

Parte da avaliação de um projecto arquitectónico é feita com base em critérios matemáticos. Estes critérios impõem limites mínimos e máximos às áreas dos vários tipos de espaços, tendo em atenção o tipo de projecto que é avaliado. Por exemplo, uma biblioteca deve ter no mínimo uma certa percentagem de área de circulação. Os critérios são basicamente condições matemáticas que devem ser verdadeiras.

Os métodos que este tipo de aplicação tipicamente utiliza são somas de áreas de serviço, circulação e servidas e o cálculo de caminhos mais curtos (por exemplo entre os extremos do projecto).

Algumas das questões importantes seriam:

- Qual a área total dos espaços de circulação ?
- Existe algum espaço com área inferior a $3m^2$?

5.2 Edifícios de uma área urbana

O objecto BlueBase projecto define uma cidade ou parte de uma cidade, em que cada espaço representa um edifício.

5.2.1 Consultar informação de recenseamentos de uma cidade

Outro tipo de aplicações que pode beneficiar deste sistema são as aplicações que associam representações gráficas urbanas (como plantas de cidades) com base de dados provenientes de recenseamentos (como os inquéritos dos censos). Para a utilização deste sistema podem ser associados os dados da base de dados aos espaços que representam edifícios. São associados a cada edifício os dados relativos à idade e estado de conservação do imóvel, os dados dos residentes, se possuem água e luz, etc.

Resposta sob a forma de representação gráfica - As perguntas seguintes resultariam no destacar ou colorir de edifícios que cumpram as condições:

- Edifícios onde residem crianças entre os 6 e os 8 anos;
- Colorir os edifícios consoante o número de residentes em cada edifício (para efeitos de densidade populacional).

Nesta aplicação são cruzados dados provenientes de recenseamentos com informação gráfica. A visualização da distribuição etária da população poderiam demostrar a necessidade de construção de equipamento social, como escolas, lares de terceira idade, pavilhões desportivos, bibliotecas e outros.

Outra hipótese é testar possíveis relações existentes entre variáveis. Por exemplo, se a existência de população não activa tem influência no estado de conservação dos imóveis que habitam.

5.2.2 Visualizar os espaços para aluguer de uma cidade

O sistema é composto por uma base de dados que descreve os vários espaços disponíveis para aluguer ou venda de uma cidade. Podem ser destacados espaços com base em critérios como preço, localização, tamanho, etc.

Algumas das consultas possíveis são:

- Espaços para aluguer com valor inferior a xxxx escudos;
- Espaços para aluguer com três ou mais quartos;
- Espaços para vender com área superior a xm^2.

Este sistema poderia ser disponibilizado numa página de Internet, de forma a permitir consultar os espaços a alugar por uma empresa do ramo.

5.3 Sub-regiões de uma grande região

Outro tipo de aplicações pretendem facultar ferramentas de trabalho sobre representações de grandes áreas. O objectivo deste tipo de representações é visualizar as várias partes que formam um todo.

Este sistema poderia representar:

- Freguesias de um concelho;
- Distritos de um país;
- Países de um continente;

Associada à representação gráfica existe um base de dados que contém as propriedades de cada região. Algumas das propriedades podem incluir:

- População;
- Dialecto;
- Tamanho da região;

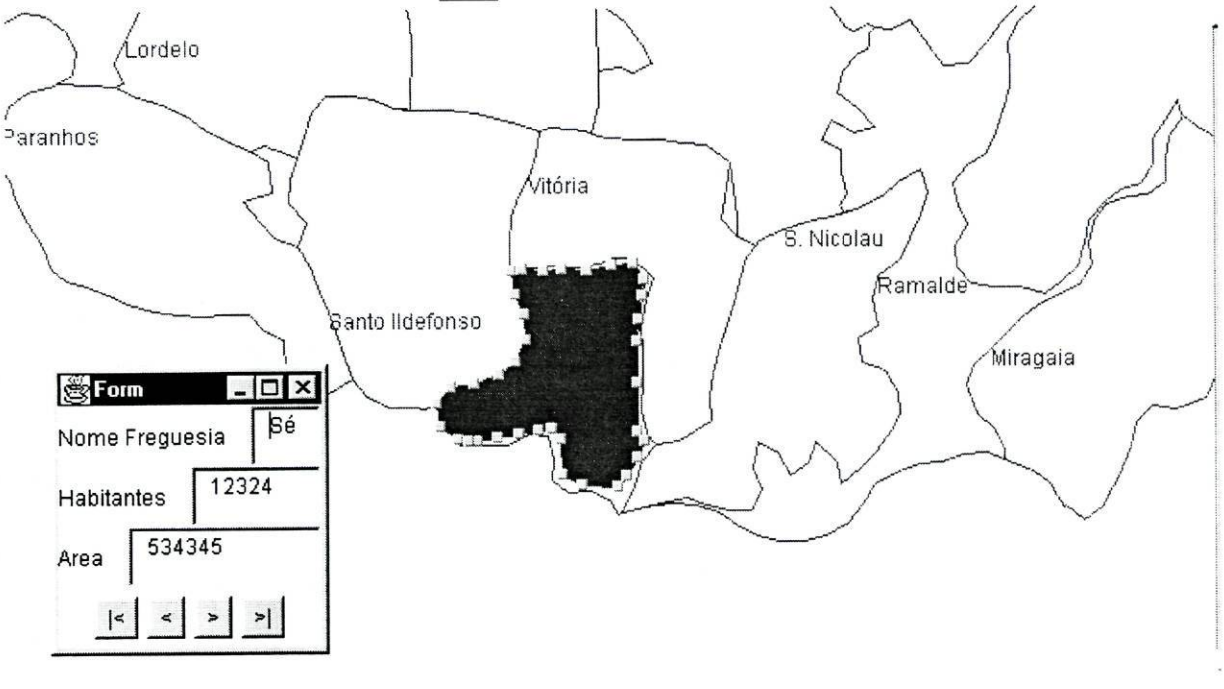


Figura 5.2: Freguesias da cidade do Porto.

Capítulo 6

Conclusão

O objectivo deste trabalho é integrar informação gráfica vectorial, dados estruturados e funcionalidades que actuam sobre a informação armazenada. A integração é implementada através da classe de objectos BlueBase. Os objectos criados desta classe são acedidos via ambiente distribuído (WWW) pelas aplicações que pretenderem os utilizar.

A classe BlueBase e restantes componentes do sistema foram implementados na linguagem Java, tirando partido da orientação por objectos, e capacidade de integração em páginas WWW.

Foi criado o utilitário BindData que visa a criação de objectos BlueBase. Este utilitário utiliza ficheiros de formato DXF com o intuito de importar a representação gráfica dos elementos armazenados nos objectos BlueBase.

Os objectos BlueBase pretendem ser interlocutores inteligentes entre as aplicações que os utilizam e as representações gráfica e estruturada que os compõem, disponibilizando um interface que apresenta ambas as representações de forma interligada.

Neste capítulo são verificados os objectivos de desenho inicialmente colocados, são analisados os problemas de implementação encontrados e é descrito algum do trabalho futuro possível.

6.1 Verificação dos objectivos de desenho

Dos objectivos de desenho iniciais apenas a integração de informação de domínio de edifícios se mostrou desadequada, tendo sido os restantes objectivos atingidos satisfatoriamente.

Criação de um formato integrado - A criação de um formato que reuna informação gráfica e informação estruturada, mostrou ser uma aproximação eficaz ao problema. Esta aproximação tem a vantagem de facilitar a utilização dos objectos pelas aplicações, dado apenas existir um interlocutor na comunicação com os dados e com as funcionalidades que com eles trabalham.

Por outro lado, a existência de um único formato facilitou a criação de métodos que trabalham com os dados estruturados e gráficos.

Utilização do sistema num ambiente distribuído - Outra vantagem advém da facilidade da sua distribuição em redes de computadores. Para esta questão muito contribuiu a simplicidade no tratamento de objectos "serializados" por parte do Java.

O ficheiro que resulta da "serialização" dos objectos pode ser convertido de forma simples num novo objecto, mesmo que este ficheiro se encontre numa máquina remota.

Por outro lado, o acesso a servidores de base de dados é igualmente simples. O pacote de objectos JDBC, incluí métodos que permitem a criação de uma conexão à base de dados remota de forma transparente.

Estas características são fruto da orientação da linguagem Java para ambientes distribuídos.

Criação de pequenas aplicações com objectivos específicos - No decorrer do trabalho foram criadas algumas aplicações com o intuito de testar as funcionalidades incluídas no objecto BlueBase.

Fruto da grande diversidade de métodos disponibilizados e do carácter alto nível na utilização destes, estas aplicações tendem a ser pequenas e simples de construir. Desta forma, grande parte da programação destas aplicações é baseada na criação de código capaz de executar as acções específicas das aplicações.

Informação de domínio relativo a edifícios - Ao contrário do que foi suposto inicialmente, a divisão dos espaços em espaços de circulação, servidos e de

serviço mostrou-se inadequada nalguns casos. Estes casos não eram inicialmente considerados por não se incluírem no campo tomado inicialmente, o da Gestão de Edifícios.

No entanto, com a generalização do pacote de objectos, novas áreas de aplicações foram sendo abarcadas pelo sistema. Aplicações que visam a representação de grandes regiões (como freguesias, distritos, países, etc.), não podem estruturar as suas áreas em função de tipos de espaços de circulação, servidos ou de serviço.

O tipo de espaço, em vez de ser utilizado como factor de organização dos espaços, deveria ser apenas uma das propriedades deste.

6.2 Problemas de implementação

Os principais problemas de implementação tiveram origem na utilização de suportes que se revelaram não ser tão "standard" quanto suposto. No entanto, é de realçar que não se encontraram outras soluções capazes de funcionar mais eficazmente que as utilizadas.

Problemas de versões da linguagem Java - Alguns problemas surgiram na execução de "applets" de Java em browsers de WWW, devido às versões de Java suportadas por estes. Contra a natureza de uma linguagem que se quer universal, existem hoje muitas versões do Java, não estando ainda bem definido um "standard" para a linguagem.

Por este facto, foi necessária a utilização de métodos que já caíram em desuso em novas versões do Java, para que fosse possível a execução dos "applets" nos "browsers" utilizados actualmente.

Utilização de formatos "standard" - Esta questão divide-se quanto à representação gráfica e quanto ao acesso a base de dados. Relativamente às base de dados, o suporte que o Java possuí para o acesso a base de dados, o JDBC, mostrou-se robusto e capaz de uma utilização eficiente. Uma das características importantes do JDBC, é a utilização da linguagem SQL como forma de comunicar as acções a desempenhar pela base de dados.

No entanto, a utilização de ficheiros gráficos no formato DXF, levantou alguns problemas. Este formato que era suposto ser um "standard" no campo do desenho assistido por computador, contém formatação específica do "software"

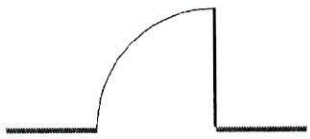


Figura 6.1: Símbolo que identifica uma porta.

que o originou. Desta forma, teve de ser definido uma versão de ficheiros DXF a ser aceite para leitura. Foi decidido aceitar a descrição de DXF gerada pelo "software" AutoCAD 14.

Implementação do algoritmo geométrico Um dos pontos importantes deste trabalho passa pelo reconhecimento de espaços fechados na representação gráfica vectorial.

O algoritmo geométrico utilizado encontrou problemas relacionados com a distribuição dos elementos pelas camadas ("layers") da representação gráfica. Cada camada está normalmente associado a um tipo de elementos. Assim, pode existir uma camada que contém as linhas que definem as paredes, outra que define as janelas, outra que define as portas, etc. No entanto, nenhuma destas camadas define completamente o interior de um espaço. Este só é definido por um polígono fechado quando são consideradas várias camadas. A utilização de uma nova camada por vezes implica que sejam consideradas novas linhas não desejadas.

A título de exemplo, vamos observar o caso das portas. É normal incluir na representação de uma porta uma curva que descreve a área percorrida pela abertura da porta (figura 6.1). Esta linha impede o algoritmo de encontrar o verdadeiro limite do espaço. Sendo as portas necessárias para encontrar um espaço fechado, a sua representação aumenta a dificuldade de encontrar esse mesmo espaço.

6.3 Trabalho futuro

Após terem sido criadas as ferramentas básicas de interligação entre informação gráfica e informação estruturada, podem surgir outras funcionalidades mais evoluídas geradas a partir das primeiras.

Ligações entre objectos - A ligação de objectos consiste na existência de uma associação entre dois objectos com alguma propriedade em comum. Por exemplo,

a uma determinada sala de um edifício pode estar associado o gabinete do responsável por essa sala. Assim, poderiam ser visualizados todos os objectos associados com o objecto actual.

A associação pode ser vista de um modo mais largo, permitindo que sejam associados dois quaisquer objectos. Desta forma, seria possível a ligação entre si de espaços, de blocos e de objectos BlueBase. O objecto BlueBase de um determinado edifício poderia estar interligado ao objecto BlueBase de outro, permitindo ao utilizar "saltar" rapidamente da visualização de um para outro.

As associações também poderiam ser feitas entre objectos de tipos diferentes. Uma determinada área podia estar associada a outro objecto BlueBase. Seria possível a criação de um conjunto de objectos BlueBase, sendo cada um especializado num nível de detalhe particular.

Por exemplo, um dos objectos BlueBase teria a seu cargo a descrição de uma cidade. Neste são visualizadas as freguesias dessa mesma cidade. A cada freguesia corresponde uma área que disponibiliza ao utilizador um conjunto de dados. Ao clicar sobre uma das áreas que define uma freguesia em particular, é possível visualizar uma planta dessa freguesia. Nesta planta, estão descritas ruas e edifícios. Alguns dos edifícios estão por sua vez associados à planta interna dos mesmos. Esta aplicação, permite ao utilizador "navegar" pela cidade através de um sistema de níveis de detalhe. A cada área estão associados outros projectos que correspondem a um nível de detalhe superior.

Inserção de imagens "raster" - Outra característica possível seria a colocação de imagens "raster" como papel de fundo da representação gráfica vectorial. Desta forma, poderia ser utilizadas imagens como meros indicadores. Tal como já foi referido na discussão dos sistemas GIS, a representação da dados que variam de forma contínua (como altitude, vegetação, condições climatéricas) é mais fácil utilizando imagens "raster" que vectoriais.

Desta forma, uma fotografia de satélite poderia servir de fundo a uma aplicação que pretenda visualizar características naturais e políticas de uma região. O utilizador de tal aplicação obteria uma representação gráfica que lhe indica a localização dos cursos de água, sendo também visualizadas as fronteiras dos países.

A representação gráfica vectorial pode não ser visualizada na aplicação, sendo utilizada simplesmente na selecção de objectos. Isto é, o utilizador não se apercebe da existência de uma camada transparente que se encontra sobre a

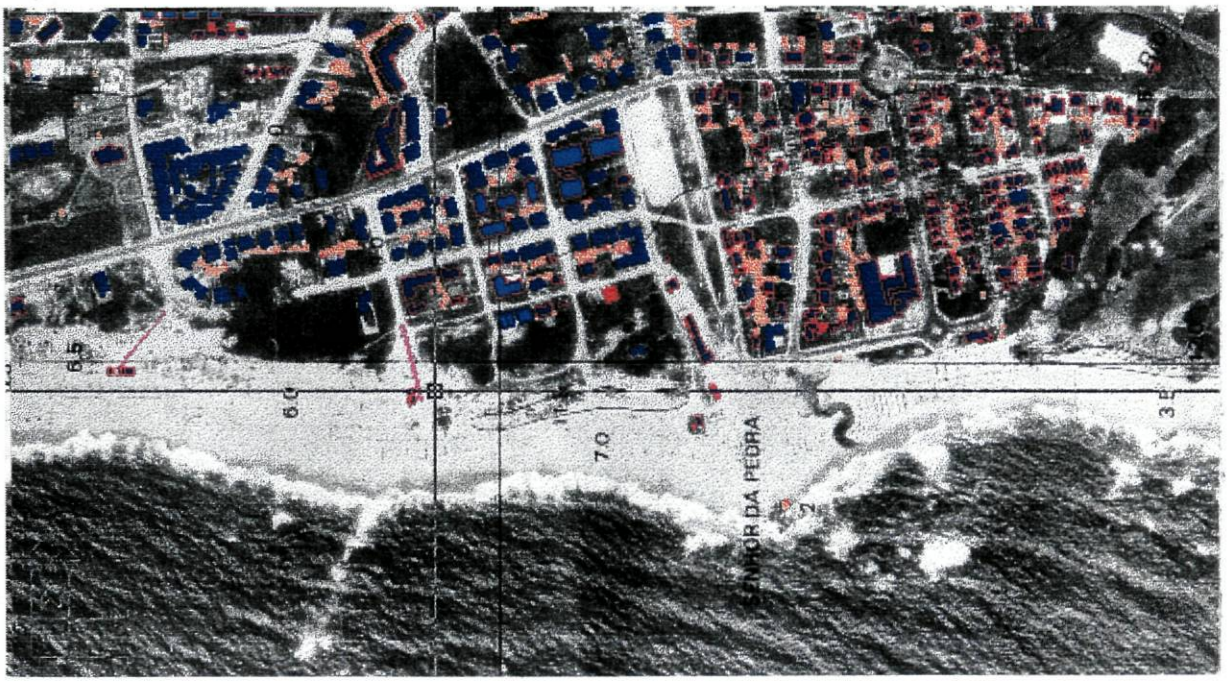


Figura 6.2: Fotografia aérea do S. da Pedra com vectorização de alguns edifícios.

imagem de fundo. A selecção de uma qualquer objecto é executado quando o utilizador "clica" na imagem de fundo.

Colorir por propriedades - Esta funcionalidade permite que os objectos sejam representados de acordo com alguma das suas propriedades.

Os espaços de um edifício poderiam ser representados utilizando padrões que identificavam o tipo de soalho que possuem, ou o departamento a que pertencem. Neste caso seriam associadas cores a diferentes valores, ou intervalos de valores, de uma dada propriedade.

Outra alternativa seria a de que uma propriedade numérica alterasse a intensidade da cor do objecto. Por exemplo, os edifícios de uma cidade poderiam ser coloridos tendo em atenção o número de crianças que nestes habitam. Desta forma, seria possível concluir da densidade de população jovem de uma cidade de forma intuitiva.

Visualização de cruzamento de variáveis - Por vezes pretende-se visualizar numa única representação gráfica o resultado de várias consultas diferentes. Nestas consultas pretendem-se destacar os elementos que respeitam mais que um critério.

Uma das possíveis formas de implementar o cruzamento de consultas, seria

proceder à alteração de cores sempre que um objecto respeitasse um novo critério. Assim, a cor (ou padrão) do objecto indicavam as suas características.

A título de exemplo, suponhamos que se pretende concluir da existência ou não de uma implicação entre a existência de água canalizada nos edifícios e o estado de conservação dos mesmos. Suponhamos que após ter sido feito a primeira consulta, os edifícios que não tem água canalizada são coloridos a vermelho. Na segunda consulta, os edifícios que estão em mau estado de conservação são coloridos a azul. Os edifícios que não tiverem água canalizada e estiverem em mau estado de conservação são coloridos a magenta, indicando que cumprem ambos os critérios.

A utilização de padrões e de transparências na representação dos objectos pode ser implementada utilizando as funcionalidade disponibilizadas na nova versão 1.2 do Java. Esta versão permite um melhoramento gráfico significativo quando comparada com versão utilizada neste trabalho.

Espaços formados por áreas não contíguas - Alguns dos espaços que se pretenderam reconhecer eram na realidade formados por áreas não adjacentes. Um dos exemplos é a divisão de um continente em países. O espaço de um país não pode ser representado por um único polígono, se o país for composto, por exemplo, por ilhas. Neste caso, seria necessário criar vários espaços.

O sistema actual não suporta a criação de espaços que não possam ser representados por único polígono fechado. A implementação de áreas não adjacentes como um único espaço, poderia ser representado não por um polígono mas sim por um conjunto de polígonos.

Apêndice A

Classes de Objectos

O conjunto das classes foi dividido em três diferentes pacotes de objectos ("packages"). Cada "package" está especializado num único tema. Os "packages" existentes são: project, geometry e quadtree.

Cada "package" é constituído por várias classes.

A descrição de cada classe consiste na enumeração dos valores que armazena bem como dos métodos que podem ser invocados.

As classe criadas utilizam alguns elementos da linguagem Java que são explicados na seguinte lista:

Hashtable - implementação em Java das tabelas de partição

ResultSet - este objecto permite o acesso a uma tabela de dados

A.1 "package" project

Este "package" é compostos pelas classes que armazenam os dados de cada projecto.

A.1.1 Classe BlueBase

Esta é a principal classe do sistema. Os objectos que descrevem os vários elementos existentes num projecto são armazenados nos objectos desta classe.

Esta classe é serializável, o que permite que seja armazenada num ficheiro.

Variáveis

- `blockDefList` - `Hashtable` de objectos do tipo `BlockDef`, que representam os vários tipos de blocos que se podem encontrar no projecto. A chave da `Hashtable` é o nome do tipo de bloco.
- `blockList` - `Hashtable` de objectos do tipo `Block`, que representam os vários de blocos que se encontram no projecto. A chave da `Hashtable` é o nome do bloco.
- `blocksTable` - Nome da tabela que contém os dados dos blocos.
- `blocksKey` - Nome do campo da tabela de blocos que contém os nomes dos blocos.
- `bounds` - Define o rectângulo de determina os limites do projecto.
- `circulationAreas` - `Hashtable` de objectos do tipo `Area`, que representam os espaços de circulação que se encontram no projecto. A chave da `Hashtable` é o nome do espaço.
- `circulationTable` - Nome da tabela que contém os dados dos espaços de circulação.
- `circulationKey` - Nome do campo da tabela de espaços de circulação que contém os nomes dos espaços.
- `data` - Lista do tipo `PropertyList`, que armazena as eventuais propriedades que se pretendam atribuir ao projecto.
- `gContext` - Objecto do tipo `GraphicsContext`, que mantém a informação relativa ao estado actual de visualização do projecto.
- `medX` - Coordenada x do ponto médio da representação gráfica do projecto.
- `medY` - Coordenada y do ponto médio da representação gráfica do projecto.
- `name` - Nome do projecto.
- `pointsPerCircle` - Número de vértices que têm os polígonos que substituem as elipses.

- `servedAreas` - `Hashtable` de objectos do tipo `Area`, que representam os espaços servidos que se encontram no projecto. A chave da `Hashtable` é o nome do espaço.
- `servedTable` - Nome da tabela que contém os dados dos espaços servidos.
- `servedKey` - Nome do campo da tabela de espaços servidos que contém os nomes dos espaços.
- `servingAreas` - `Hashtable` de objectos do tipo `Area`, que representam os espaços que servem que se encontram no projecto. A chave da `Hashtable` é o nome do espaço.
- `servingTable` - Nome da tabela que contém os dados dos espaços que servem.
- `servingKey` - Nome do campo da tabela de espaços que servem que contém os nomes dos espaços.

Métodos

- `add` - Adiciona uma propriedade.
- `addBlock` - Adiciona um bloco ao projecto.
- `addBlock` - Cria um novo bloco com um nome.
- `addBlockDef` - Cria um novo tipo de bloco com um nome.
- `addCirculationArea` - Adiciona um espaço de circulação.
- `addCirculationArea` - Cria um espaço de circulação com um nome.
- `addCirculationToDB` - Cria um registo de espaço de circulação na base de dados.
- `addServedArea` - Adiciona um espaço servido.
- `addServedArea` - Cria um espaço servido com um nome.
- `addServedToDB` - Cria um registo de espaço servido na base de dados.
- `addServingArea` - Adiciona um espaço de serviço.
- `addServingArea` - Cria um espaço de serviço com um nome.
- `addServingToDB` - Cria um registo de espaço de serviço na base de dados.

- `areaInCanvasXY` - Procura uma área em (x,y) da tela utilizada na visualização.
- `areaInXY` - Procura uma área em (x,y) da representação interna.
- `colorOfType` - Retorna a cor associada a um tipo de espaço.
- `draw` - Desenha todos os elementos tendo em consideração um contexto gráfico.
- `findPathBetween` - Retorna o caminho (se existir) entre duas áreas.
- `getAllBlocks` - Retorna uma lista com todos os blocos existentes na base de dados.
- `getAllCirculation` - Retorna uma lista com todas as áreas de circulação existentes na base de dados.
- `getAllServed` - Retorna uma lista com todas as áreas servidas existentes na base de dados.
- `getAllServing` - Retorna uma lista com todas as áreas de serviço existentes na base de dados.
- `getAllTableNames` - Retorna os nomes das tabelas existentes na base de dados.
- `getBlock` - Retorna um bloco com base no nome.
- `getBlockDef` - Retorna um tipo de bloco com base no nome.
- `getBounds` - Retorna o rectângulo que limita o projecto.
- `getCirculation` - Retorna uma área de circulação com base no nome.
- `getLeftTopCorner` - Retorna as coordenadas do canto superior esquerdo do projecto.
- `getRightBottomCorner` - Retorna as coordenadas do canto inferior direito do projecto.
- `getServed` - Retorna uma área servida com base no nome.
- `getServing` - Retorna uma área de serviço com base no nome.
- `highlightAreasPaths` - Destaca os caminhos de uma lista de áreas.
- `highlightPath` - Destaca o caminho descrito por uma lista de áreas.

- `highlightSpace` - Destaca uma área com base no nome e no tipo da área.
- `removeBlock` - Elimina um bloco do projecto.
- `removeCirculationArea` - Elimina uma área de circulação do projecto.
- `removeServedArea` - Elimina uma área servida do projecto.
- `removeServingArea` - Elimina uma área de serviço do projecto.
- `runSQL` - Executa uma instrução SQL e retorna um objecto ResultSet.
- `setDBProperties` - Determina a forma como se acede à base de dados.
- `writeToFile` - Serializa o objecto BlueBase (e todos os seus componentes) num ficheiro.

A.1.2 Classe Property List

Esta classe define uma lista de propriedades.

Métodos

- `add` - Adiciona um propriedade e respectivo valor à lista.
- `get` - Retorna as propriedades de lista com base num número.

A.1.3 Classe Area

Esta classe define uma área existe no projecto.

Variáveis

- `data` - Lista do tipo `PropertyList`, que armazena as eventuais propriedades que se pretendam atribuir a uma área.
- `name` - Nome da área.
- `pathList` - Lista de elementos `Path` que descreve as áreas com que comunica.
- `type` - Tipo de área.

Métodos

- `add` - Adiciona uma propriedade.
- `addPath` - Adiciona uma novo caminho a esta área.
- `addPoint` - Adicionar um novo ponto ao polígono que define a área.
- `addPoints` - Adicionar um conjunto de pontos ao polígono que define a área.
- `draw` - Desenha a área.
- `drawPaths` - Desenha os caminhos desta área.
- `drawSinglePath` - Desenha o caminho entre a esta área e outra área.
- `getAllPathsOrderedByDistance` - Retorna os caminhos desta área ordenado ascendentemente em relação às suas distancias em relação a uma área.
- `getPath` - Retorna, se existir, um objecto que define o caminho entre esta área e outra área.
- `highlight` - Destaca esta área.
- `removePath` - Elimina um caminho desta área.

A.1.4 Classe BlockDef

Esta classe define um tipo de bloco existente no projecto.

Variáveis

- `data` - Lista do tipo `PropertyList`, que armazena as eventuais propriedades que se pretendam atribuir a um tipo de bloco.
- `name` - Nome do tipo de bloco.
- `polygons` - Lista de polígonos que definem a representação gráfica deste tipo de blocos.

Métodos

- `add` - Adiciona uma propriedade.
- `addFPolygon` - Adiciona um polígono (um objecto `FPolygon`) na representação deste tipo de bloco.
- `draw` - Desenha este tipo de bloco.
- `highlight` - Destaca este tipo de bloco.
- `setName` - Atribuí um nome a este tipo de bloco.

A.1.5 Classe Block

Esta classe define um bloco existente num projecto.

Variáveis

- `blockType` - Tipo de bloco.
- `boundingBox` - Rectângulo que define os limites do bloco.
- `data` - Lista do tipo `PropertyList`, que armazena as eventuais propriedades que se pretendam atribuir a um bloco.
- `name` - Nome do bloco.
- `origin` - Coordenadas que definem a localização do bloco.
- `rotation` - Ângulo que define a rotação do bloco.
- `xscale` - Escala da largura deste bloco em relação ao seu tipo.
- `yscale` - Escala da altura deste bloco em relação ao seu tipo.

Métodos

- `add` - Adiciona uma propriedade.
- `draw` - Desenha o bloco.
- `getName` - Retorna o nome do bloco.
- `highlight` - Destaca este bloco.

- `inside` - Retorna verdadeiro se os limites deste bloco incluem um coordenada.
- `setBlockType` - Associa um tipo de bloco a este bloco.
- `setName` - Atribui um nome ao bloco.
- `setOrigin` - Define as coordenas de origem.

A.1.6 Classe Path

Esta classe mantém dados relativos a um caminho entre duas áreas.

Variáveis

- `distance` - A distancia deste caminho.
- `toArea` - Para que área é definido este caminho.

Métodos

- `getArea` - Retorna a área deste caminho.
- `getDistance` - Retorna a distancia deste caminho.

A.1.7 Classe ProjCanvas

Esta classe tem o objectivo de facilitar a visualização de um objecto da classe BlueBase. Esta classe define uma tela que pode conter elementos visuais como barras de deslocamento. Suporta zoom e deslocamentos horizontais e verticais. Mantém uma ou mais áreas e um blocos seleccionado, e é contem uma barra de estado.

Variáveis

- `currentAreas` - Áreas que se encontram seleccionadas, e por essa razão destacadas na visualização.
- `gContext` - Contexto gráfico (zoom, deslocamento horizontal e vertical), objecto do tipo `GraphicsContext`.
- `hasScrollBar` - Verdadeiro se existem uma barras de deslocamento.

- `projToDraw` - Mantém o objecto `BlueBase` visualizado nesta tela.
- `statusBar` - Barra de estado com informação sobre os elemento seleccionados.

Métodos

- `clickedOn` - Método invocado se houve um clique na tela.
- `doubleClickedOn` - Método invocado se houve um duplo clique na tela.
- `runSQL` - Executa uma consulta SQL à base de dados e retorna um objecto `ResultSet`.
- `setProject` - Associa um objecto `BlueBase` a esta tela.
- `setScrollBars` - Associa barras de deslocamento à tela.
- `viewAll` - Altera o zoom de forma a ser visualizado todo o projecto.
- `viewPan` - Visualiza a área definida por uma rectângulo.

A.1.8 Classe GraphicsContext

Esta classe define mantém informação relativa a um contexto gráfico.

Variáveis

- `currentTool` - A ferramenta actual.
- `graphics` - A tela gráfica.
- `size` - O tamanho do projecto.
- `xpan` - O deslocamento horizontal.
- `ypan` - O deslocamento vertical.
- `zoom` - O zoom actual.

Métodos

- `getCanvasSize` - Retorna o tamanho da tela.

- `getCurrentTool` - Retorna o índice da ferramenta actual.
- `setCanvasSize` - Actualiza o tamanho da tela.
- `setCurrentTool` - Actualiza a ferramenta actual.
- `setParams` - Actualiza zoom, deslocamento horizontal e vertical.

A.1.9 Classe DataBaseConnection

Esta classe define um conexão a uma base de dados.

Variáveis

- `port` - Porta do servidor a base de dados a aceder.
- `protocol` - Protocolo a utilizar no acesso à base de dados.
- `url` - Endereço a utilizar no acesso à base de dados.

Métodos

- `addCirculation` - Adiciona um registo de área de circulação.
- `addServed` - Adiciona um registo de área servida.
- `addServing` - Adiciona um registo de área de serviço.
- `close` - Fecha a conexão à base de dados.
- `connect` - Tenta estabelecer a conexão à base de dados.
- `getAllTableNames` - Retorna uma lista de tabelas existentes na base de dados.
- `runQuery` - Executa uma consulta à base de dados.
- `updateArea` - Actualiza uma área da base de dados.

A.2 "package" geometry

Este "package" inclui um conjunto de classes geométricas.

A.2.1 Classe FPoint

Esta classe define um ponto com coordenadas do tipo `float`.

Variáveis

- `x` - Coordenada x do ponto.
- `y` - Coordenada y do ponto.

Métodos

- `angle` - Retorna o ângulo formado pelo vector (0,0)-(x,y).
- `equals` - Retorna verdadeiro se dois pontos são iguais.
- `norm` - Retorna a norma do vector (0,0)-(x,y).
- `pointOf` - Retorna verdadeiro se este ponto está no interior de um rectângulo.
- `toPoint` - Retorna um objecto do tipo `Point` (definido na API do Java).

A.2.2 Classe FLine

Esta classe define um segmento de recta com coordenadas do tipo `float`.

Variáveis

- `origin` - Coordenada da origem do segmento de recta.
- `destin` - Coordenada do destino do segmento de recta.

Métodos

- `angle` - Inclinação do segmento.
- `intersect` - Calcula a intersecção entre dois segmentos.
- `norm` - A norma deste segmento.

A.2.3 Classe FRectangle

Esta classe define um rectângulo com coordenadas do tipo `float`.

Variáveis

- `x1` - Coordenada x do canto superior esquerdo.
- `y1` - Coordenada y do canto superior esquerdo.
- `x2` - Coordenada x do canto inferior direito.
- `y2` - Coordenada y do canto inferior direito.

Métodos

- `inside` - Verdadeiro se o rectângulo inclui um dado ponto.

A.2.4 Classe FPolygon

Esta classe define um polígono com coordenadas do tipo `float`.

Variáveis

- `npoints` - O número de vértices que compõe o polígono.
- `xpoints` - A lista das coordenadas x dos vértices.
- `ypoints` - A lista das coordenadas y dos vértices.

Métodos

- `addPoint` - Adiciona um vértice ao polígono.
- `change` - Retorna outro `FPolygon` aplicando transformações geométricas de zoom, e deslocamento ao actual.
- `getBoundingBox` - Retorna um rectângulo que define a área rectangular coberta pelo polígono.
- `inside` - Retorna verdadeiro se um dado ponto se encontra no interior do polígono.
- `toPolygon` - Retorna um objecto do tipo `Polygon` (definido na API do Java).

A.3 "package" quadtree

Este "package" define uma única classe que faz a divisão de segmentos de recta e de blocos por cinco áreas. Estas cinco áreas são quatro quadrantes de dimensões idênticas e uma quinta área que abrange os quatro quadrantes.

A.3.1 Classe QuadTree

Variáveis

- `allBlocks` - Lista da totalidade de blocos.
- `allLines` - Lista da totalidade de segmentos de recta.
- `allQuad` - Lista de quadrantes que mantêm as linhas.
- `allQuadBlocks` - Lista de quadrantes que mantêm os blocos.

Métodos

- `calculateQuad` - Calcula o quadrante de um ponto.
- `getAllBlocks` - Retorna uma lista de todos os blocos.
- `getBlockIn` - Retorna o bloco existente num dado ponto.
- `getQuad` - Retorna o quadrante de um bloco ou linha.
- `getVectorOfQuad` - Retorna uma lista dos segmentos existentes num dado quadrante.
- `getVectorOfQuadBlock` - Retorna uma lista dos blocos existentes num dado quadrante.

Referências

[ACAD14] AutoCAD Release 14 documentation, Autodesk

[GHRC] Graham Hamilton e Rick Cattel, JDBC: A Java SQL API, JavaSoft

[HS90] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990

[JGH96] James Gosling Henry McGilton: The Java Language Environment, 1996

[JUJW97] Jeffrey D. Ullman e Jennifer Widom: A First Course in Database Systems, Prentice Hall, 1997

[MP] Mukesh Prasad: Intersection of Line Segments, Geographics Gems II, Academic Press

[SQL92] ANSI X3, Database Language SQL, 1992